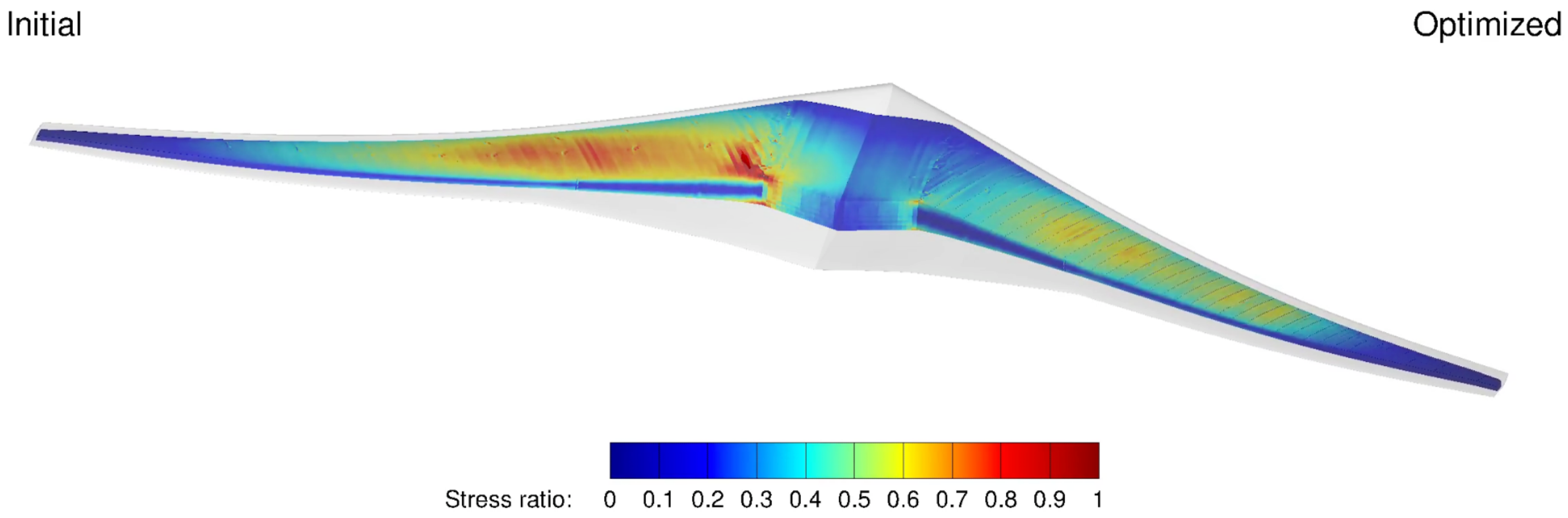
FUNtoFEM: A Framework for High-Fidelity Aeroelastic Analysis and Design



Kevin Jacobson
Aeroelasticity Branch, NASA LaRC

LaRC/Northrop Grumman TIM
May 10, 2019

Motivation

NASA CFD 2030 Vision [1]

- Outlines what the state-of-the-art CFD should be by 2030
- **Grand Challenge Problem #3**:
  - Multidisciplinary analysis and design (MDAO) of a highly flexible advanced aircraft configuration
  - Goal: demonstrate the ability to perform CFD-based system level optimization of an advanced configuration that **requires both steady and unsteady high-fidelity models**
  - “… including **explicit aeroelastic constraints that may require a time-accurate CFD approach**”
- One of the major impediments to MDAO is the “lack of sensitivity information for optimization and uncertainty quantification”

[1]

[1] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis, “CFD Vision 2030 Study: a Path to Revolutionary Computational Aerosciences,” NASA, Tech. Rep. CR-2014-218178, 2014.

Motivation

- Traditionally, aeroelastic analysis with FUN3D done in two ways:
  1. Transfer of loads and displacements to external structures code through file I/O
  2. Incorporate structural solver into FUN3D 'nodet' executable
     - Internal modal structural solver
     - Solver specific interfaces for rotorcraft analysis (Dymore, CAMRAD, RCAS)
- Difficult to maintain all those interfaces
- General aeroelastic interface added to FUN3D:
  - Treat FUN3D as a component in the multidisciplinary analysis rather than the driver
  - Set surface node displacements and/or rigid transform matrices
  - Get surface forces
- FUNtoFEM – more general coupling of FUN(3D) to FEM
  - Python-based aeroelastic driver – steady and time accurate analysis
  - FEM-based aeroelastic analysis and adjoint-based optimization

FUN3D - Mesh Motion

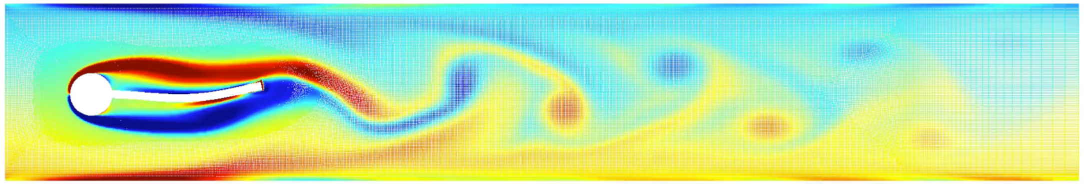
- Types of mesh motion:
  - 'deform' mesh motion:
    - Linear elasticity model displaces volume mesh nodes

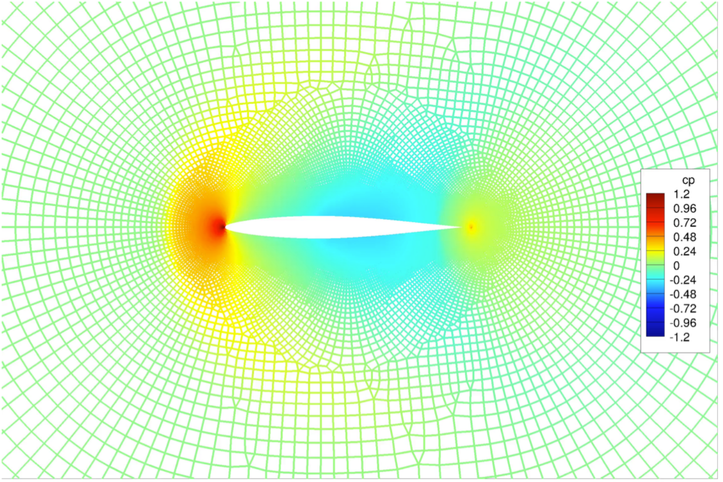
$$\mathbf{x}_{A0} + \mathbf{u}_A - \hat{\mathbf{x}}_A - \mathbf{K}_G \left(\mathbf{x}_G - \hat{\mathbf{x}}_G \right) = 0$$

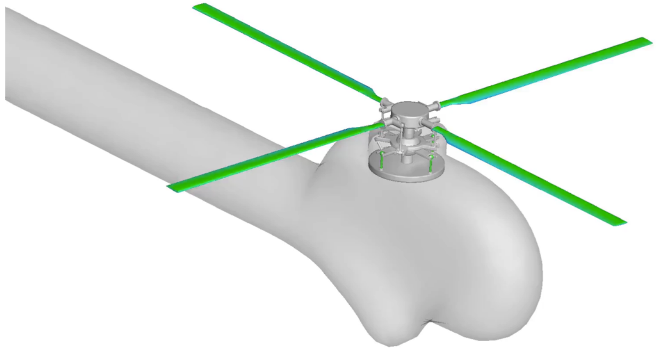
  - 'rigid' mesh motion:
    - Transformation matrix moves the volume mesh

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

  - 'rigid+deform' motion:
    - Reduce amplitude of deformation
    - Combine with overset meshes
      - Large relative motion







FUN3D - Modifications for FUNtoFEM

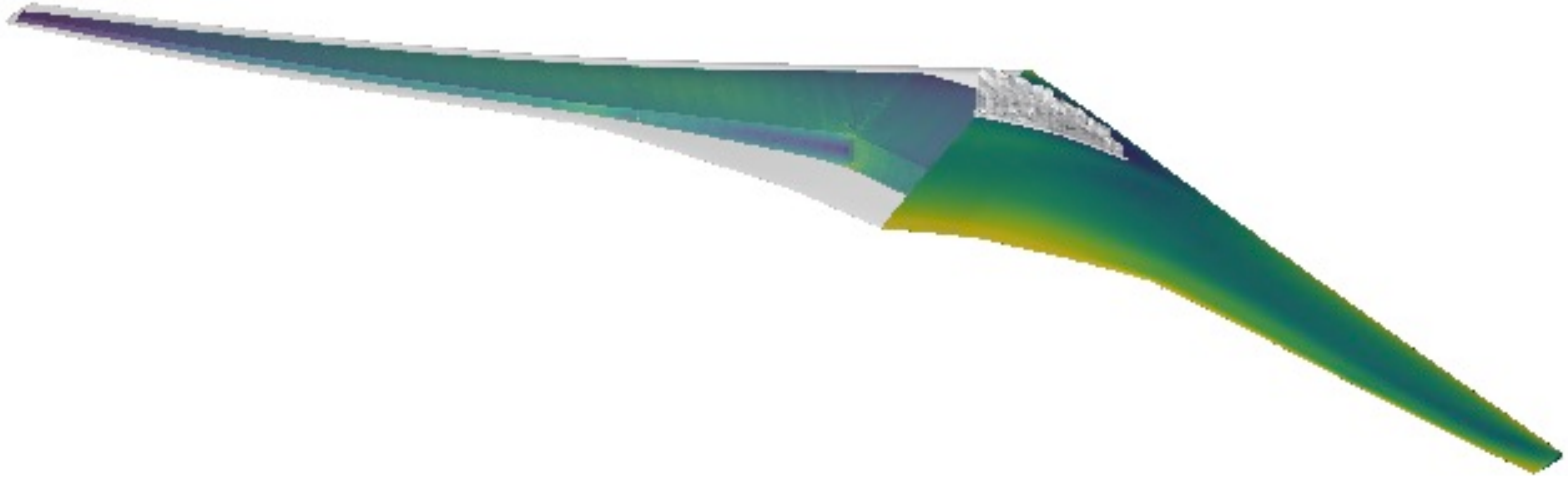
Aeroelastic Python interface

- Body motion input
  - Rigid transform
  - Surface deformation
- Force integration and extraction
  - Pressure and skin friction forces
- Corresponding adjoint interfaces
  - Limited to compressible path
  - no skin friction
- Not required to use FUNtoFEM to utilize this interface

```python
flow = Flow()
flow.initialize_project()
flow.initialize_data()
flow.initialize_grid()
flow.initialize_solution()

#=============================================================================
# Pull out the surface node information for coupling
#=============================================================================
ibody = 1 # fun3d numbering so starts at 1
num_nodes = flow.extract_surface_num(body=ibody)
x,y,z     = flow.extract_surface(num_nodes,body=ibody)

#=============================================================================
# Time step loop
#=============================================================================
for step in range(prior_steps+1,prior_steps+1+nsteps):
    if 'rigid' in mesh_movement:
        transform = get_rigid_motion(step)
        flow.input_rigid_transform(transform,body=ibody)
    if 'deform' in mesh_movement:
        dx, dy, dz = get_deform_motion(step,x,y,z)
        flow.input_deformation(dx,dy,dz,body=ibody)
    flow.iterate()

    fx, fy, fz = flow.extract_forces(num_nodes,body=ibody)

#=============================================================================
# Finish the fun3d analysis. Write restart files, hist files, etc
#=============================================================================
flow.post()
```

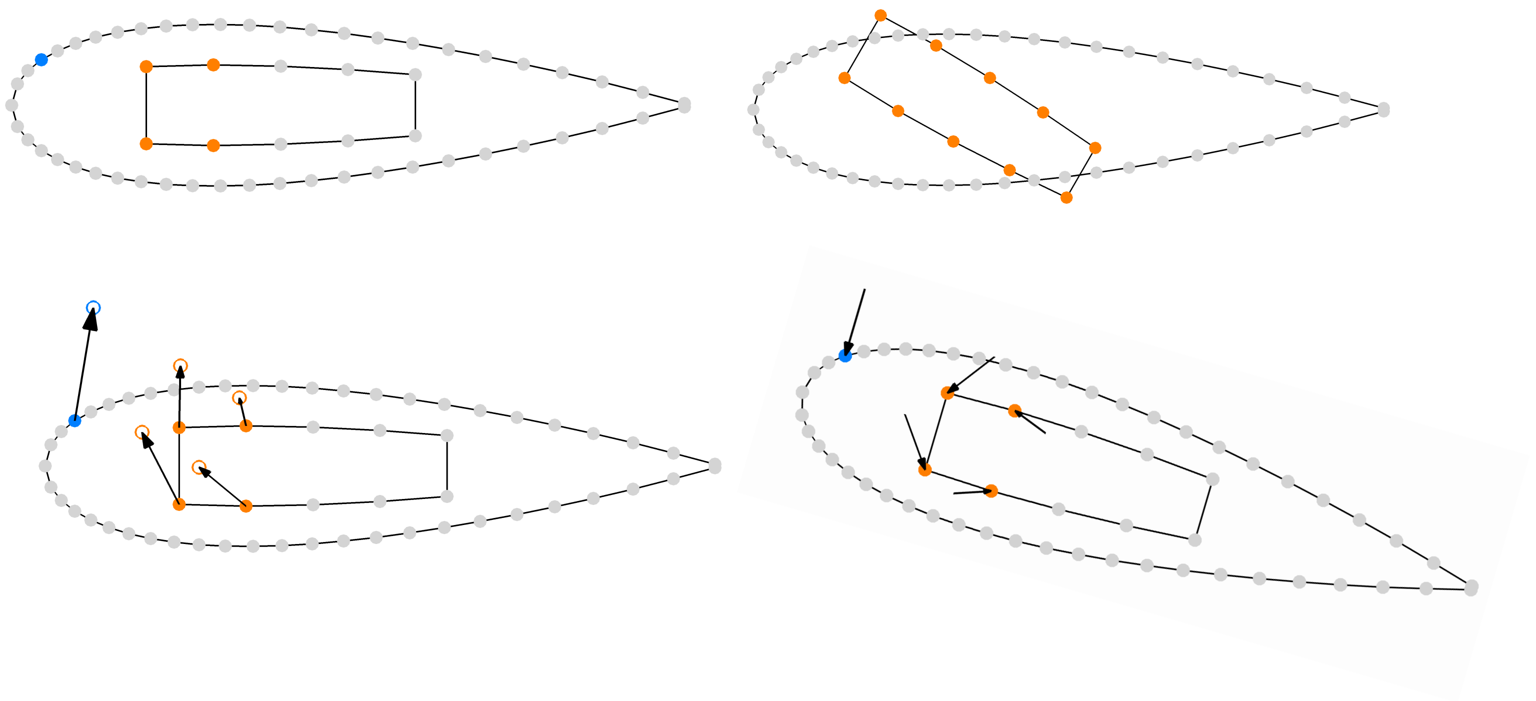
FUNtoFEM

- Transfer schemes for load and displacement transfers
- Python-based aeroelastic driver for analysis and optimization
  - Steady and time-domain aeroelasticity
  - Multibody and multipoint optimization
  - Adjoint-based sensitivities
    - Of any function defined in the disciplinary solvers
    - With respect to any design variable defined in the disciplinary solvers
    - With respect to shape/planform variables that affect structures and aerodynamics



FUNtoFEM - Transfer Schemes

- Matching-based Extrapolation of Loads and Displacements (MELD)
  - Displacement Transfer
    - Attach each aerodynamic node to weighted centroid of N nearest structural nodes
    - Least-squares problem for best fit motion of the structural nodes
    - Discretization independent
  - Load Transfer
    - Principle of virtual work



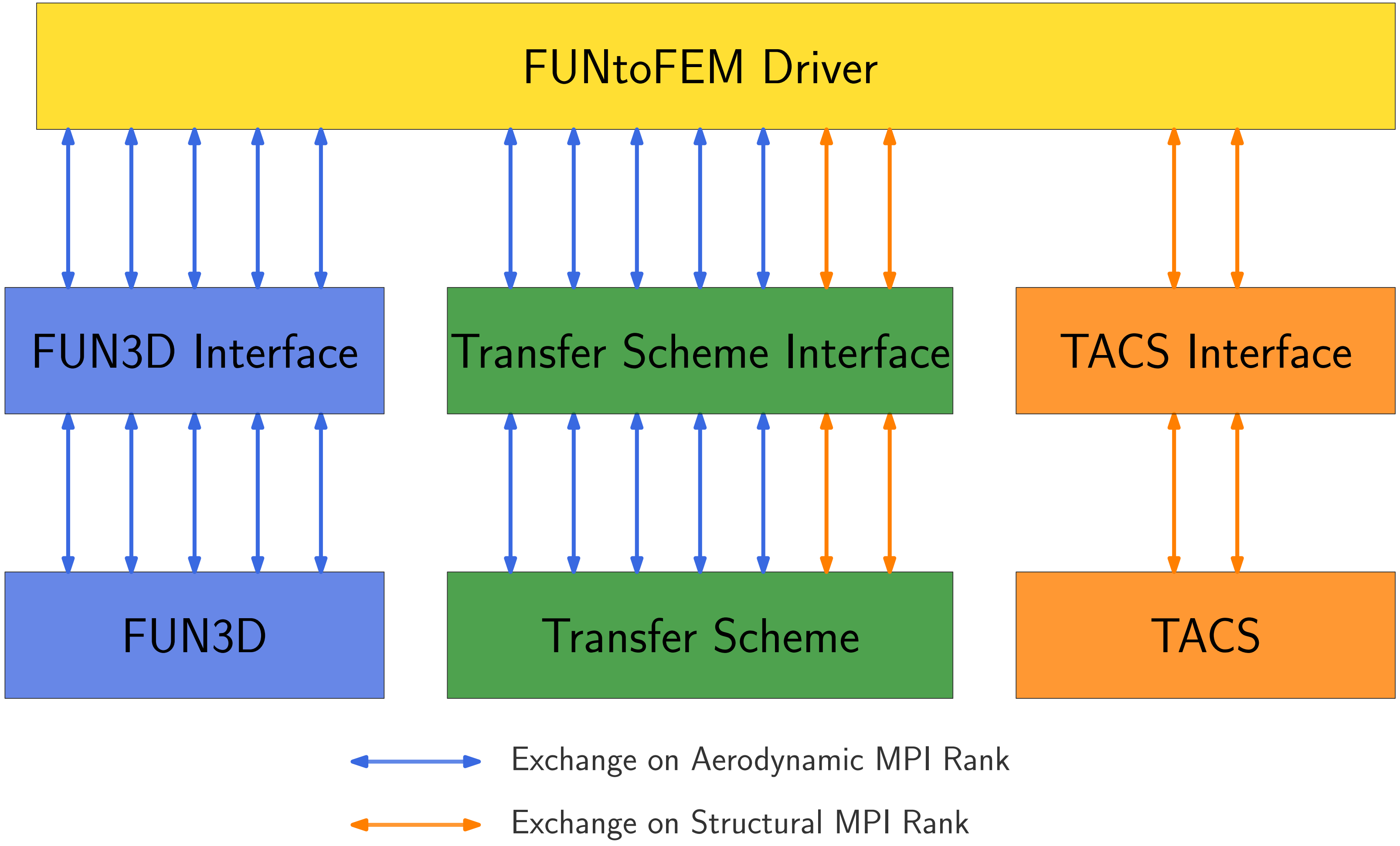
FUNtoFEM - Transfer Schemes

- Motion decomposition (rigid+deform motion)
  - Rigid motion extraction:
    - Utilize best fit motion (least-squares) kernel from MELD
  - Elastic motion extraction:
    - Local frame deformation

$$\mathbf{u}_{A*} = \mathbf{T}^{-1}(\mathbf{x}_{A0} + \mathbf{u}_A - \mathbf{T}\mathbf{x}_{A0})$$

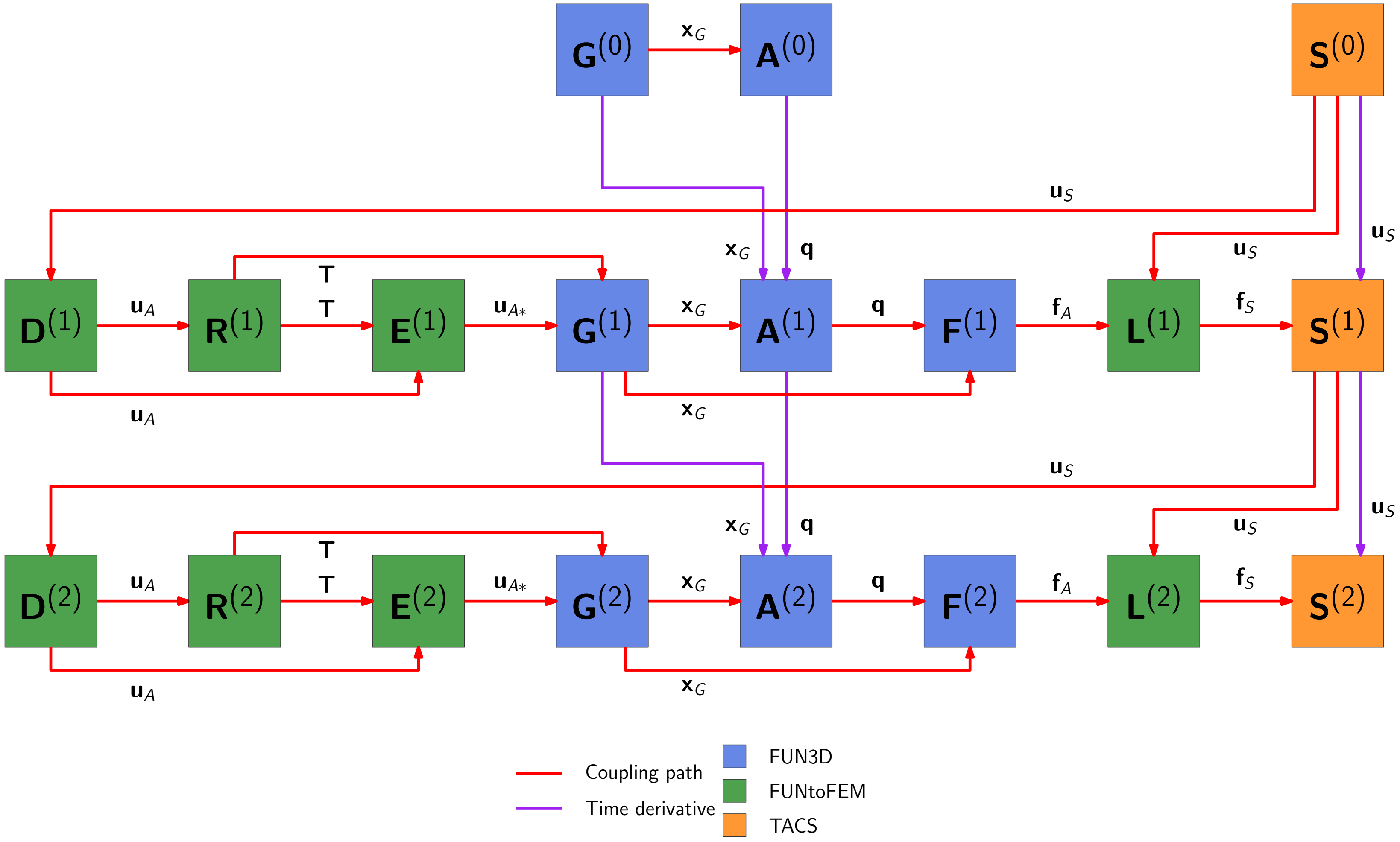
FUNtoFEM - Driver

- Modularity - swap out disciplinary components, shape parameterization, transfer schemes
  - FUN3D+TACS (FEM)
  - FUN3D+modal solver
  - CART3D+TACS
  - CART3D+modal solver
  - SU2+TACS
- In-core data transfer
- MPI-based parallelism

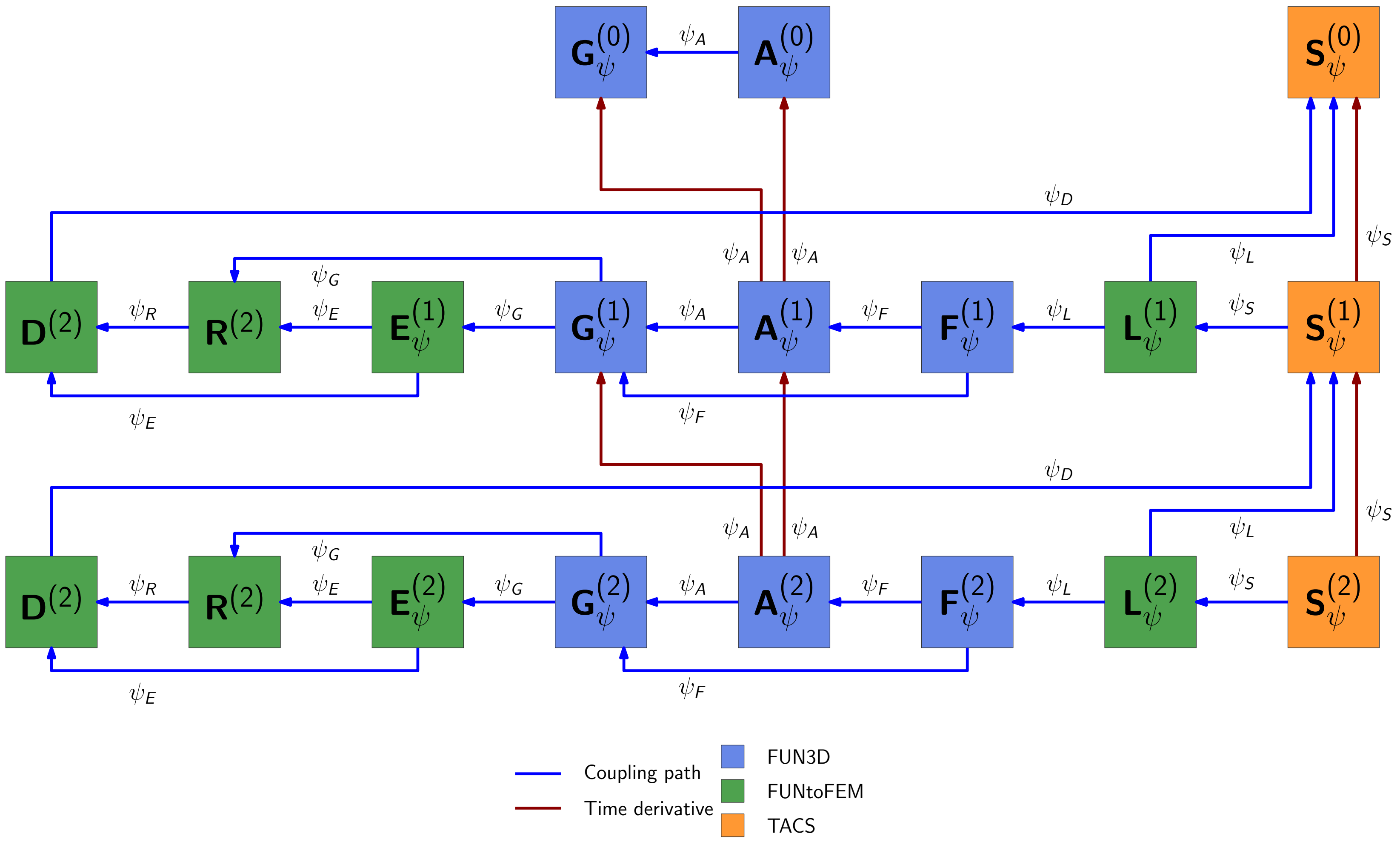


Time-accurate Coupling - Forward

Nonlinear block Gauss-Seidel Algorithm

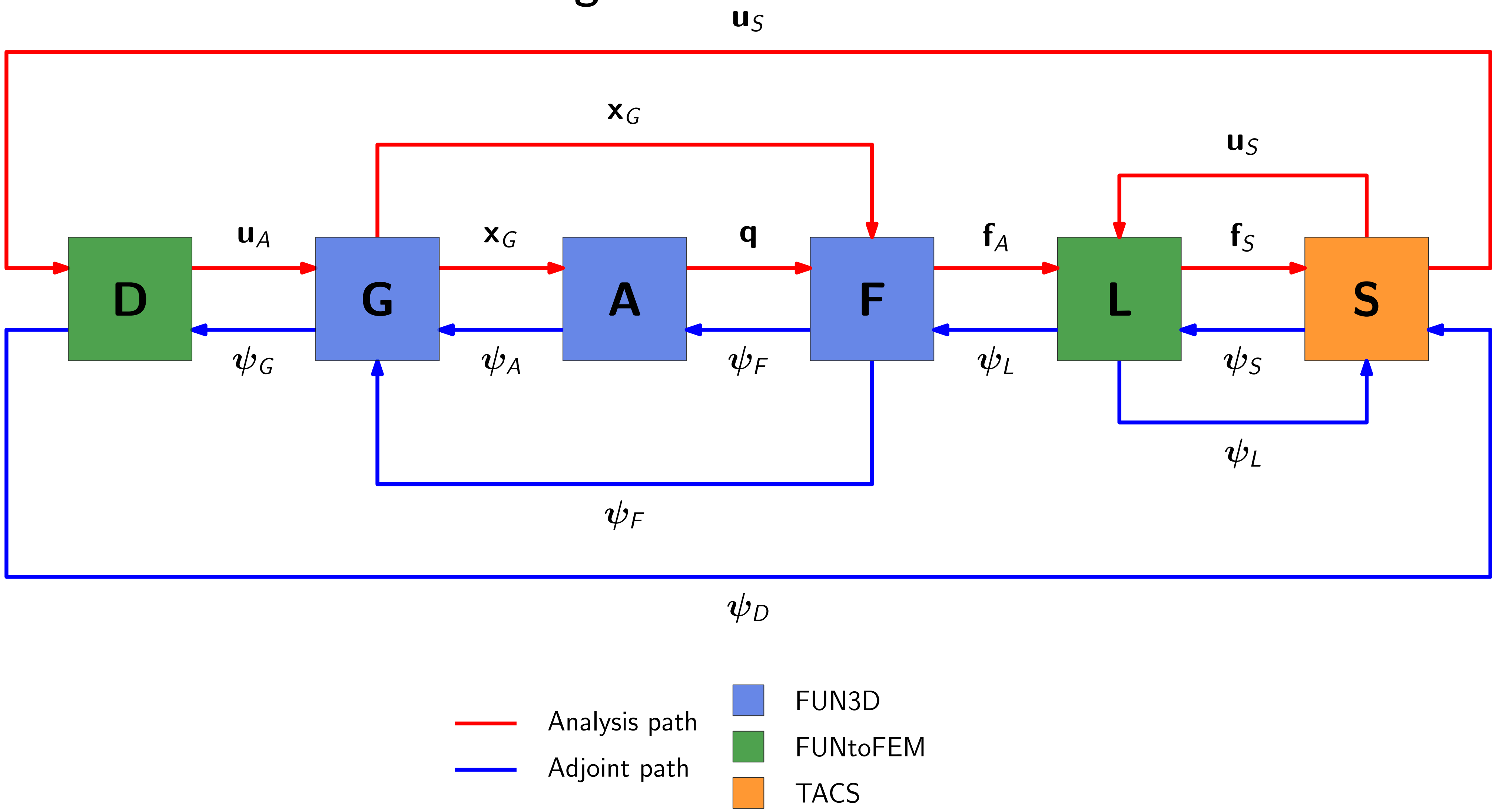


Time-accurate Coupling - Adjoint



Steady Coupling

Nonlinear block Gauss-Seidel Algorithm



Adjoint-based Sensitivity Verification

Steady:

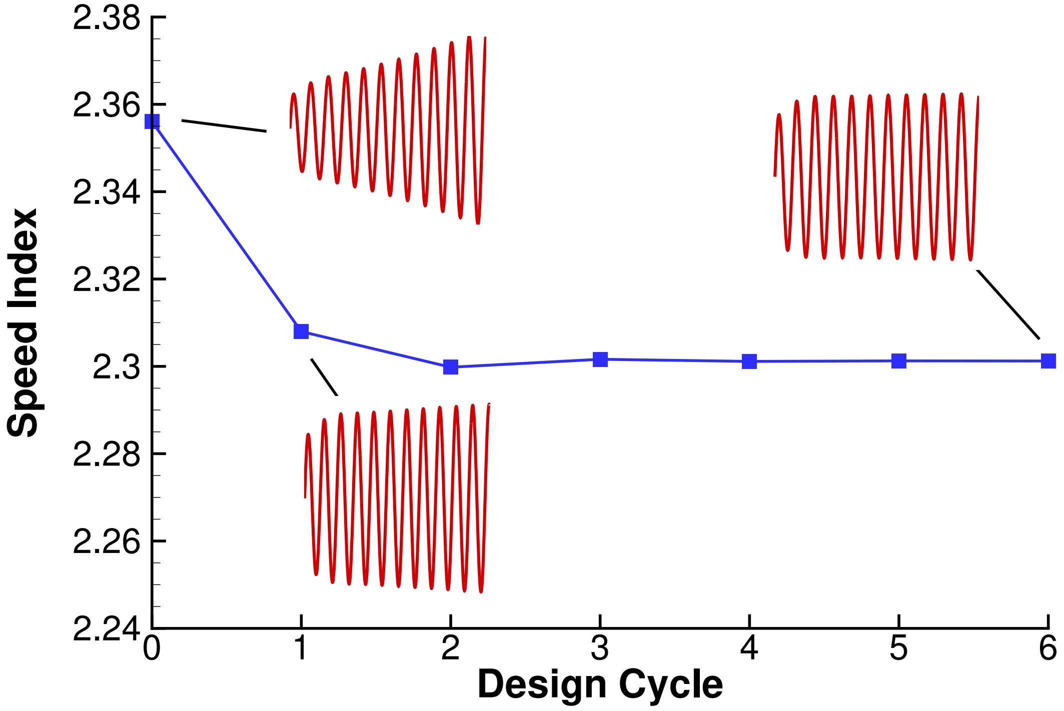
	Structural Thickness	Angle of Attack	Shape Variable
Lift Adjoint	-0.000691483665**24**	30.56134974174**8**	6.3571270080**978**
Lift Complex	-0.000691483665**10**	30.56134974174**9**	6.3571270080**967**
KS Failure Adjoint	$-1.5137829\mathbf{692}\times10^{-5}$	0.0085284874047**5**	0.0074682888033**8**
KS Failure Complex	$-1.5137829\mathbf{685}\times10^{-5}$	0.0085284874047**9**	0.0074682888033**7**

Time-accurate:

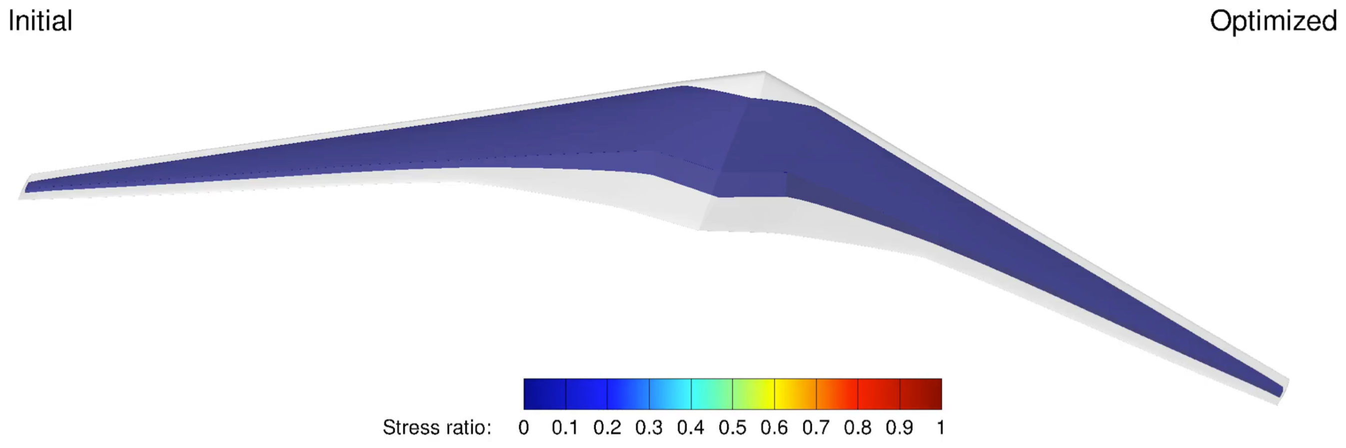
	Structural Thickness	Angle of Attack	Shape Variable
Lift Adjoint:	$8.2843976481\mathbf{3}\times10^{-7}$	$7.83592732405\times10^{-6}$	$2.409236730\mathbf{43}\times10^{-5}$
Lift Complex:	$8.2843976481\mathbf{4}\times10^{-7}$	$7.83592732405\times10^{-6}$	$2.409236730\mathbf{25}\times10^{-5}$
KS Failure Adjoint:	-0.0306817554094	$1.34592942323\times10^{-7}$	$-1.29787578524\times10^{-3}$
KS Failure Complex:	-0.0306817554094	$1.34592942323\times10^{-7}$	$-1.29787578524\times10^{-3}$

Adjoint-based Aeroelastic Optimization

- Flutter constraints
  - Automated damping calculation
  - Sensitivity of damping w.r.t. design variables and flow conditions

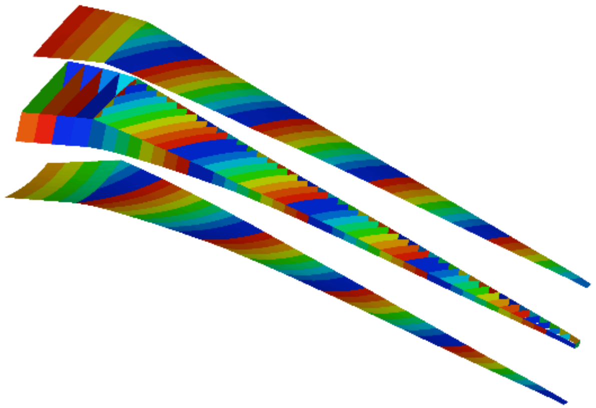


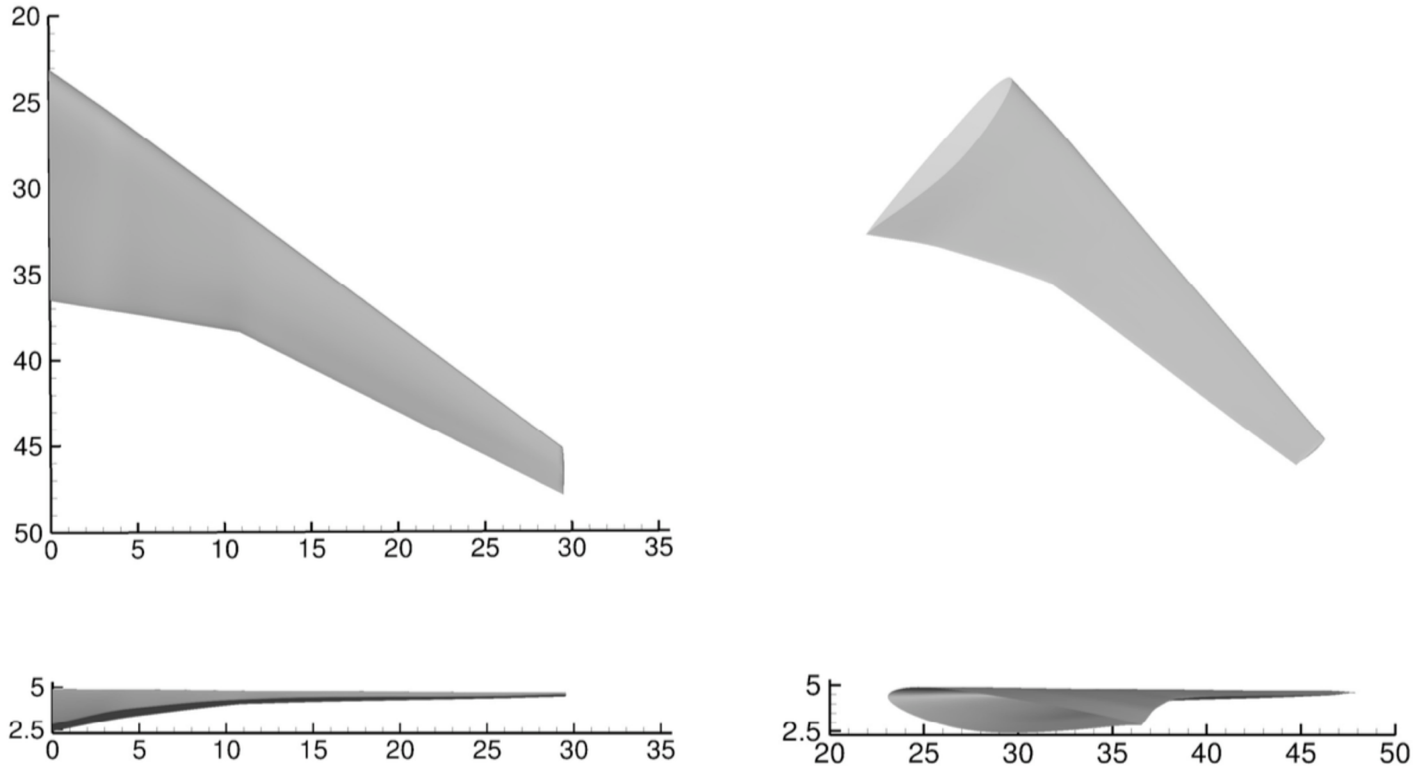
- Gust-response constraints
  - Field-velocity method for gust modeling



Undeformed Common Research Model (uCRM)

- Aeroelastic version of the NASA Common Research Model (CRM)
  - Representative of transonic commercial transport aircraft
  - Reverse engineered jig shape OML and wingbox to match the original CRM [1]
- Wingbox:
  - Ribs
  - Leading and trailing edge spars
  - Upper and lower skins





[1] G. Kenway, G. Kennedy, and J. Martins, “Aerostructural Optimization of the Common Research Model Configuration,” in *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2014

uCRM Optimizations

- Maneuver-constrained takeoff gross weight (TOGW) minimization
  - Stress constraint – 2.5 G pullup maneuver constraint (steady analysis)
  - Analysis – 2 steady
- Gust-constrained mass minimization:
  - Stress constraint – cruise gust constraint
  - Analysis – 1 time-domain

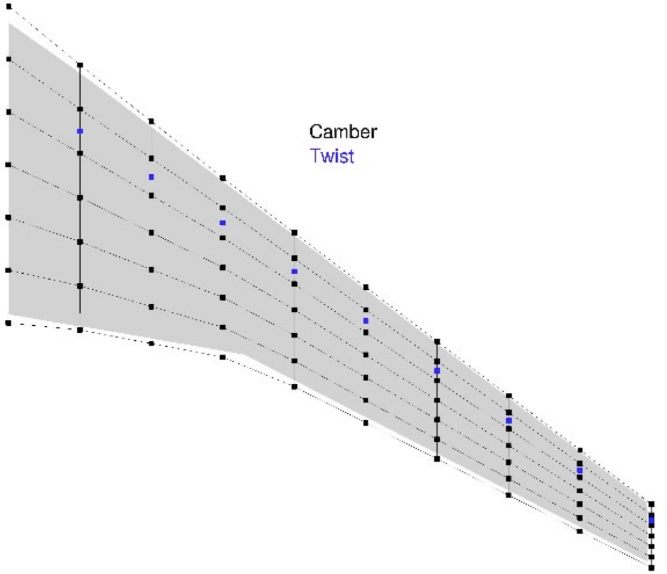
uCRM – TOGW Minimization

Computational Model:

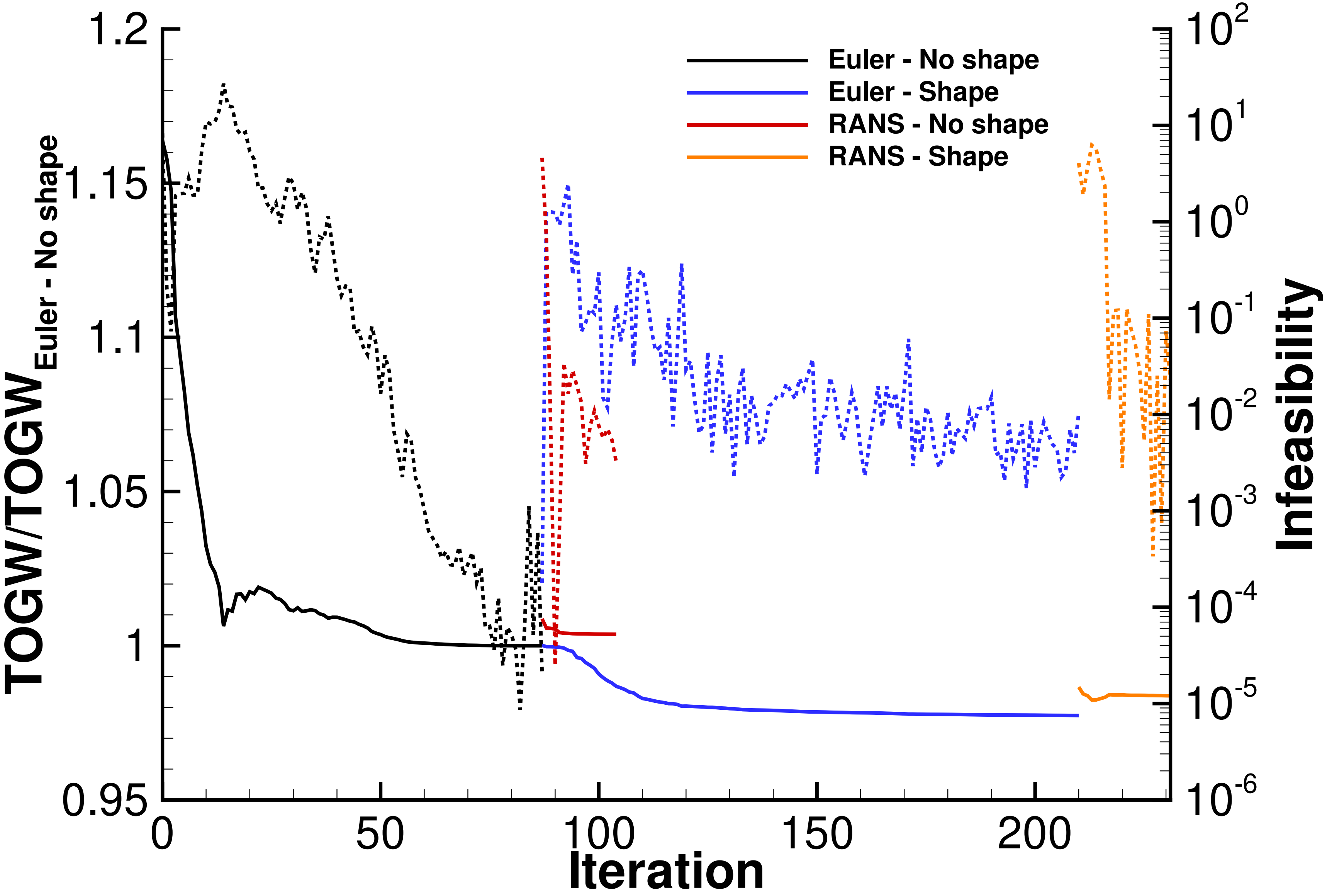
- Cruise:
  - M_∞=0.85
  - h = 35,000 ft
- Maneuver:
  - 2.5G symmetric pull-up
  - M_∞=0.86
  - h = 20,000 ft
- FUN3D:
  - Euler and RANS (SA), compressible
  - Euler - 60,742 nodes
  - RANS – 412,910 nodes
- TACS:
  - 10,584 linear shell elements

Design Problem:

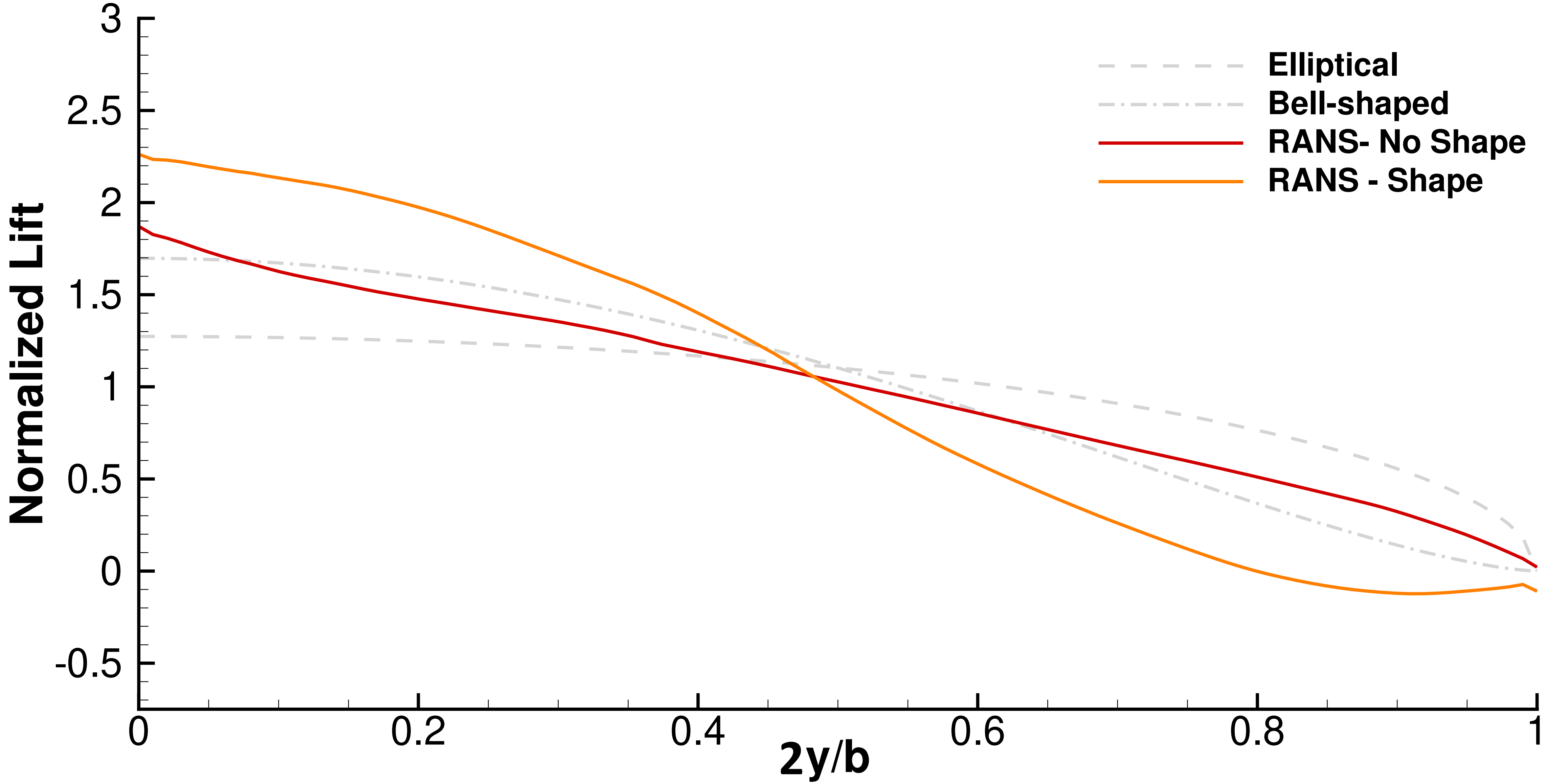
- Design variables (321):
  - Structural panel thicknesses (240)
  - Angle of attack (2)
  - Twist (9) and Camber (70)
- Objective:
  - Minimize $TOGW = \mathrm{f}\,(\mathit{empty\ weight}, L/D)$
- Constraint:
  - $1.5\ KS(\text{maneuver } \mathit{stress}) < 1$
  - Cruise: $L = W$
  - $\mathit{Maneuver}$: $L = 2.5\ W$



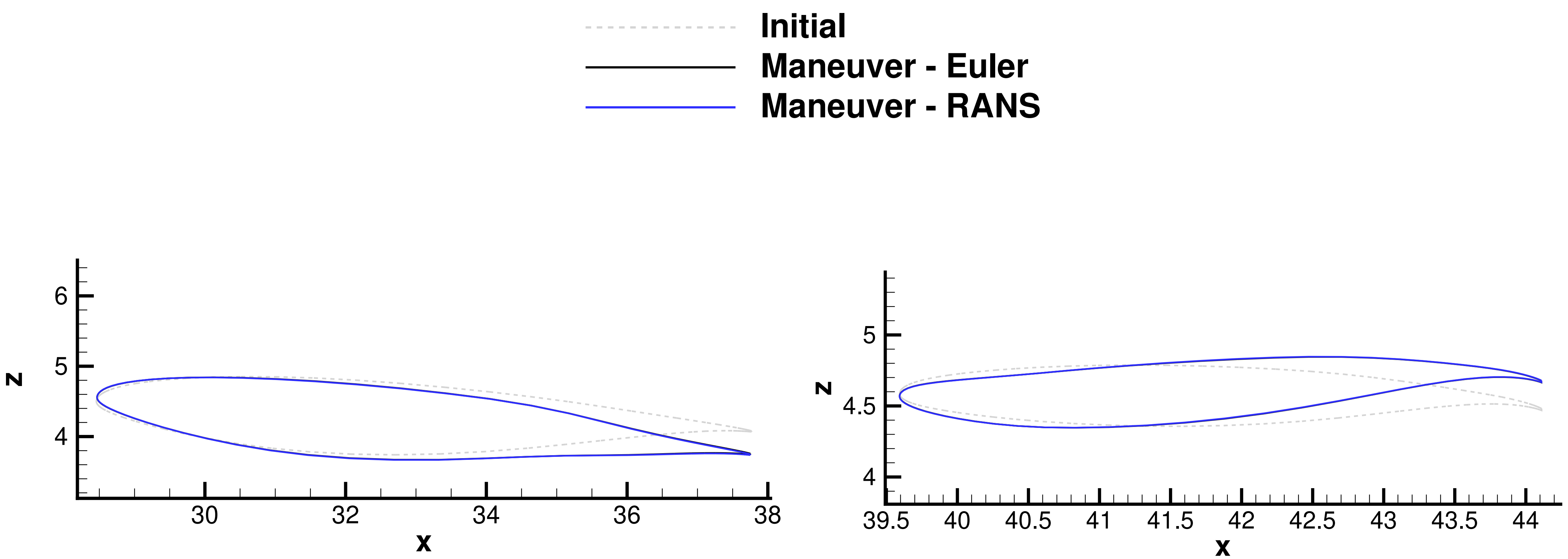
uCRM – TOGW Minimization



uCRM – TOGW Minimization



uCRM – TOGW Minimization



2y/b = 0.25

2y/b = 0.75

uCRM – Mass Minimization

Computational Model:

- Assumed Conditions:
  - M_∞=0.85, AOA = 5°
  - h = 35,000 ft
- Gust model:
  - 500 time steps
  - F_g = 0.95
  - H = 30 ft
- FUN3D:
  - Euler and RANS compressible
  - Same meshes as TOGW minimization
  - BDF2opt integration
- TACS:
  - Same mesh as TOGW minimization
  - BDF2 integration

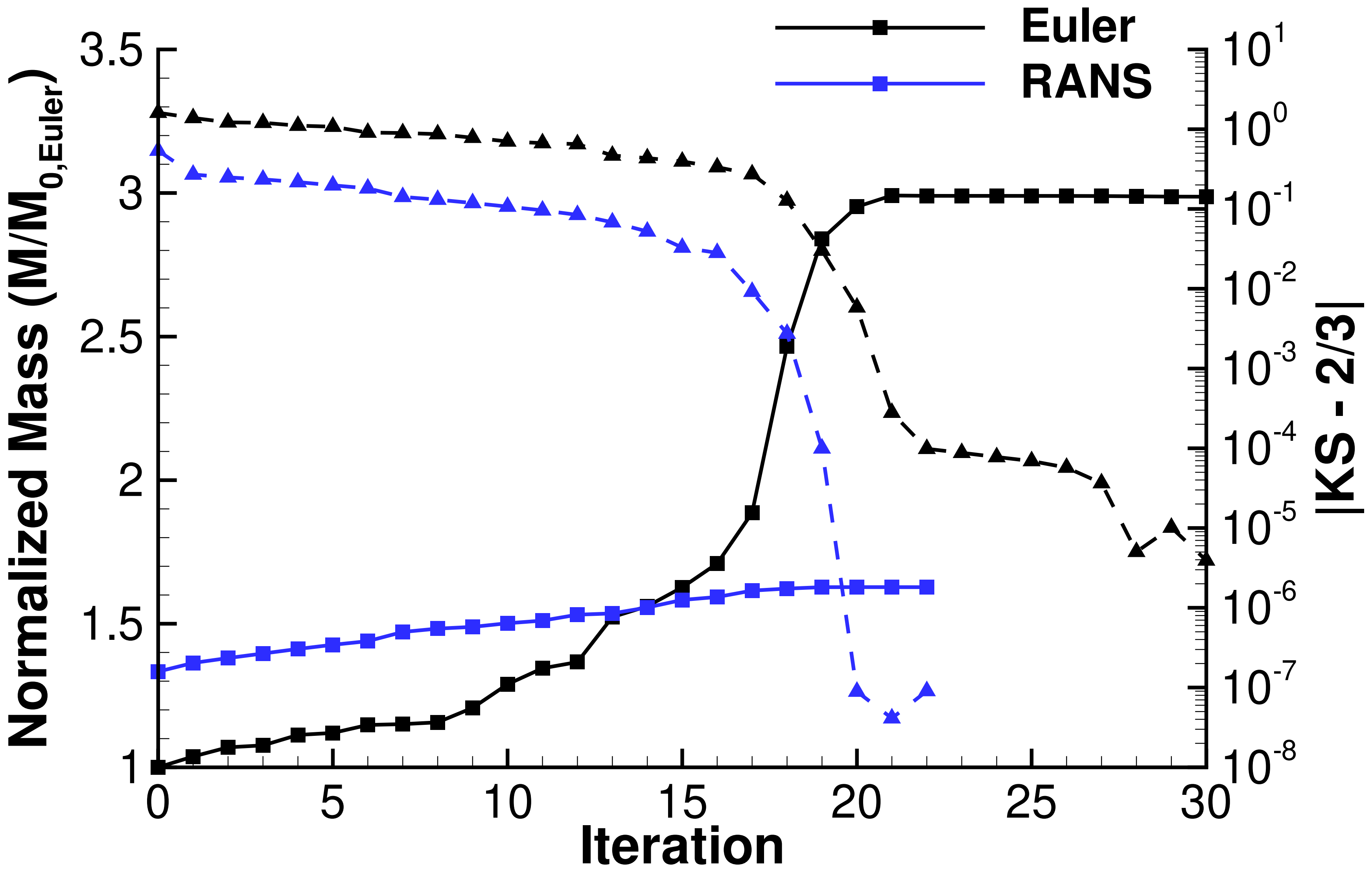
Design Problem:

- Design variables:
  - Structural panel thicknesses (240)
- Objective:
  - Minimize $mass$
- Constraint:
  - $1.5*KS(stress\ ratio) < 1$

Computational Cost:

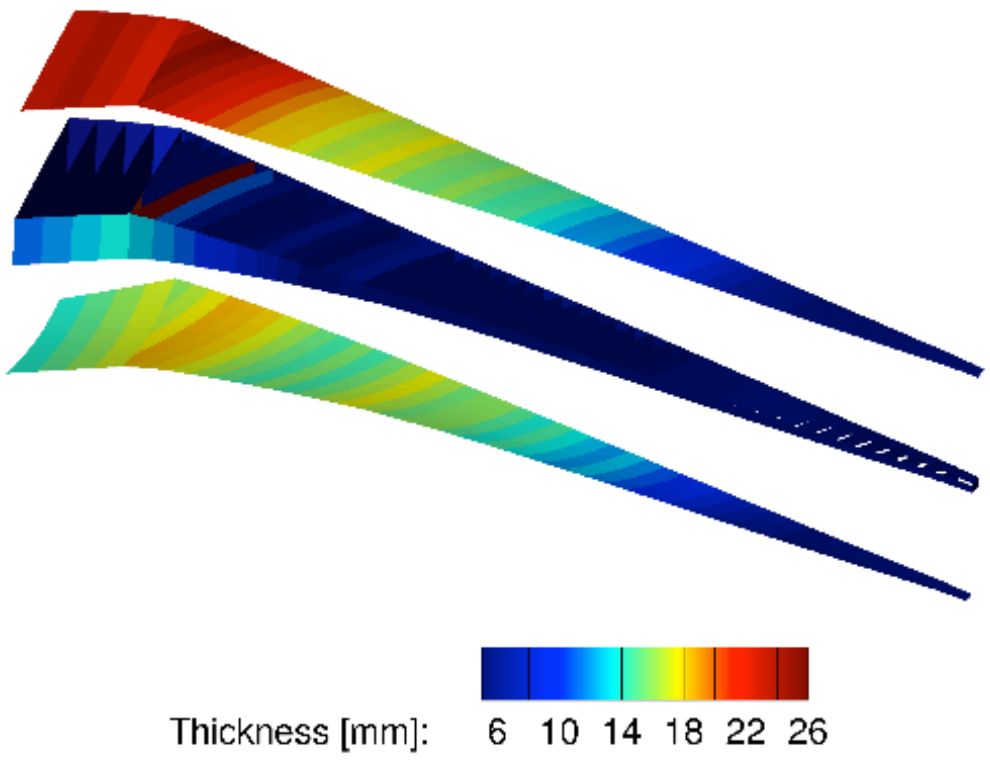
- Euler (80 cores):
  - Forward: 12 min
  - Adjoint: 20 min
- RANS (240 cores):
  - Forward: 40 min
  - Adjoint: 130 min

uCRM – Mass Minimization

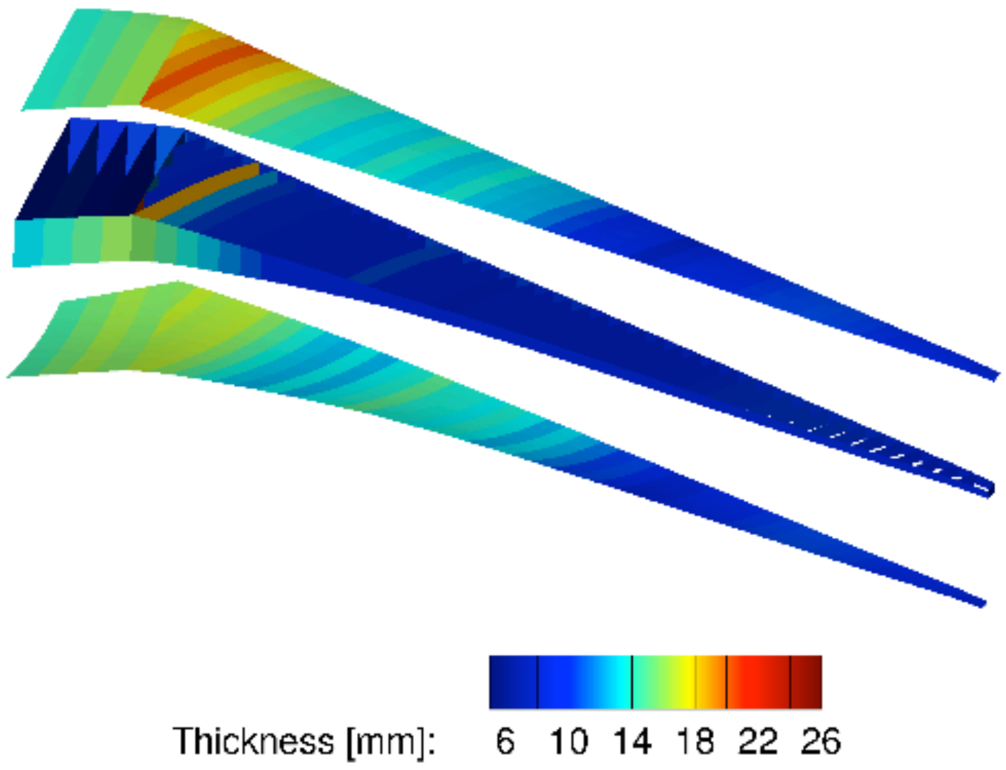


Squares with solid lines– mass, triangles with dashed lines - KS

uCRM – Mass Minimization



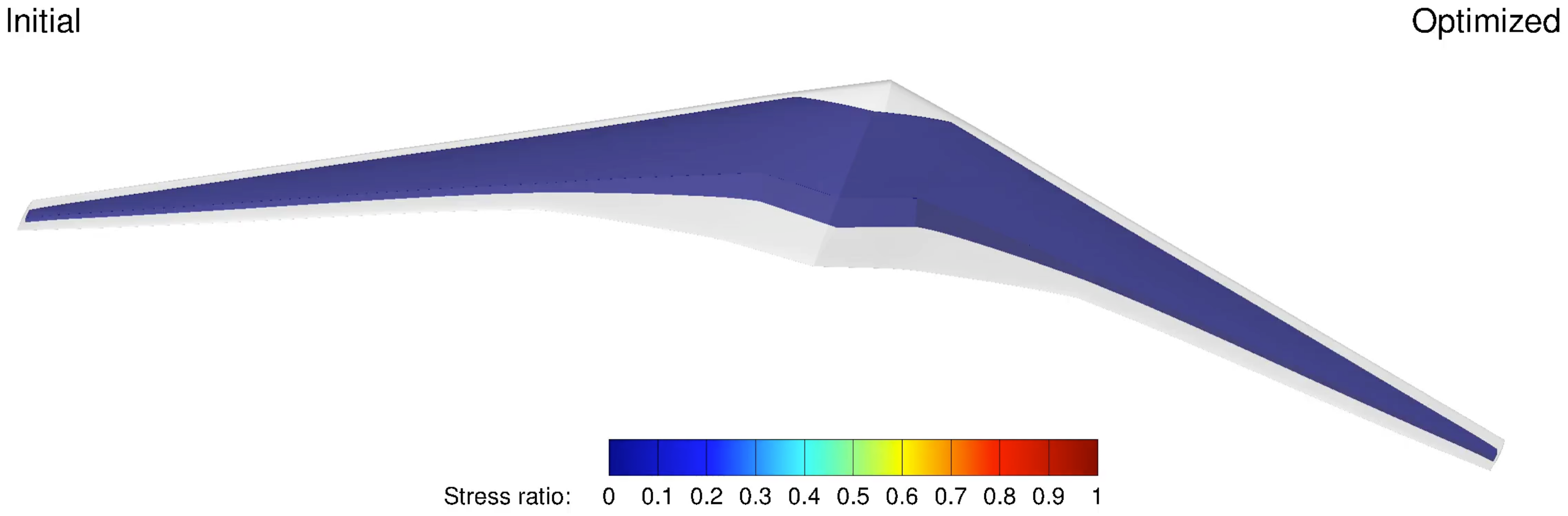
Euler



RANS

Optimized Wingboxes

uCRM – Mass Minimization



RANS Optimization

Summary

- FUN3D modifications
  - Added a more general aeroelastic interface to the Python extension module
- FUNtoFEM
  - Load and displacement transfer
  - Python-based driver for aeroelastic analysis and optimization
    - Steady and time-domain coupling

Current/Future Work:

- Integration of FUNtoFEM with OpenMDAO
- Frequency domain analysis for more efficient CFD-based flutter constraints

Publications

Publications:

- K. Jacobson, J. Kiviaho, G. Kennedy, M. Smith, “Evaluation of Time-accurate Damping Identification Methods for Flutter-constrained Optimization” *Journal of Fluids and Structures,* May 2019.
- R. Biedron, K. Jacobson, W. Jones, S. Massey E. Nielsen, B. Kleb, and X. Zhang. “Sensitivity Analysis for Multidisciplinary Systems (SAMS)” September 2018, NASA/TM–2018-220089.
- J. Kiviaho, K. Jacobson, M. Smith, and G. Kennedy, “A Robust and Flexible Coupling Framework for Aeroelastic Analysis and Optimization,” *AIAA Aviation*, Denver, Colorado, June 2017.
- K. Jacobson, J. Kiviaho, M. Smith, and G. Kennedy, “An Aeroelastic Coupling Framework for Time-Accurate Aeroelastic Analysis and Optimization,” *AIAA SciTech,* Kissimmee, Florida, January 2018.
- J. Kiviaho, K. Jacobson, M. Smith, and G. Kennedy, “Application of a Time-Accurate Aeroelastic Coupling Framework to Flutter-Constrained Design Optimization,” *AIAA Aviation*, Atlanta, Georgia, June 2018.
- K. Jacobson, J. Kiviaho, G. Kennedy, S. Massey, “A Framework for High-Fidelity Unsteady Aeroelastic Analysis and Design,” *AIAA Aviation* (presentation only), Atlanta, Georgia, June 2018

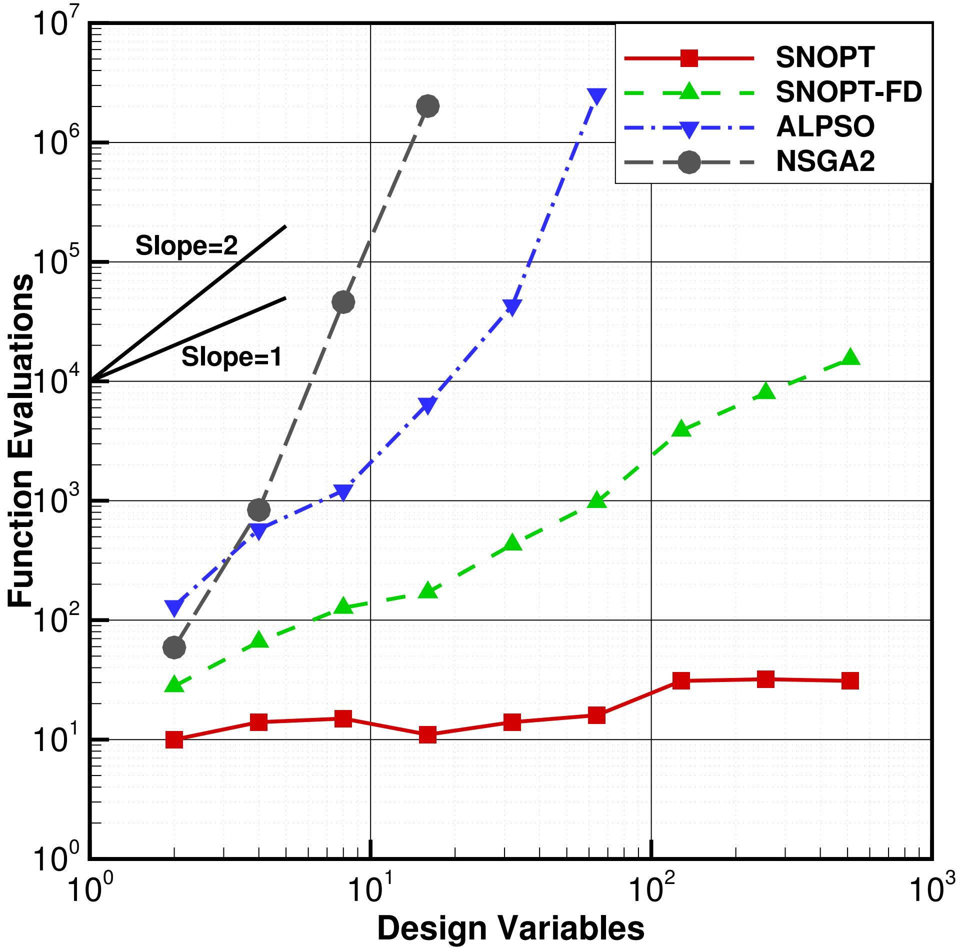
Acknowledgements:

- Funding for FUNtoFEM development was provided by NASA through the Transformative Tools and Technologies program with grant number NNX15AU22A with Technical Monitor Steve Massey
- Computational Resources supporting this work were provided by the NASA High-End Computing (HEC) Program through the NASA Advanced Supercomputing (NAS) Division at Ames Research Center

Backup Slides

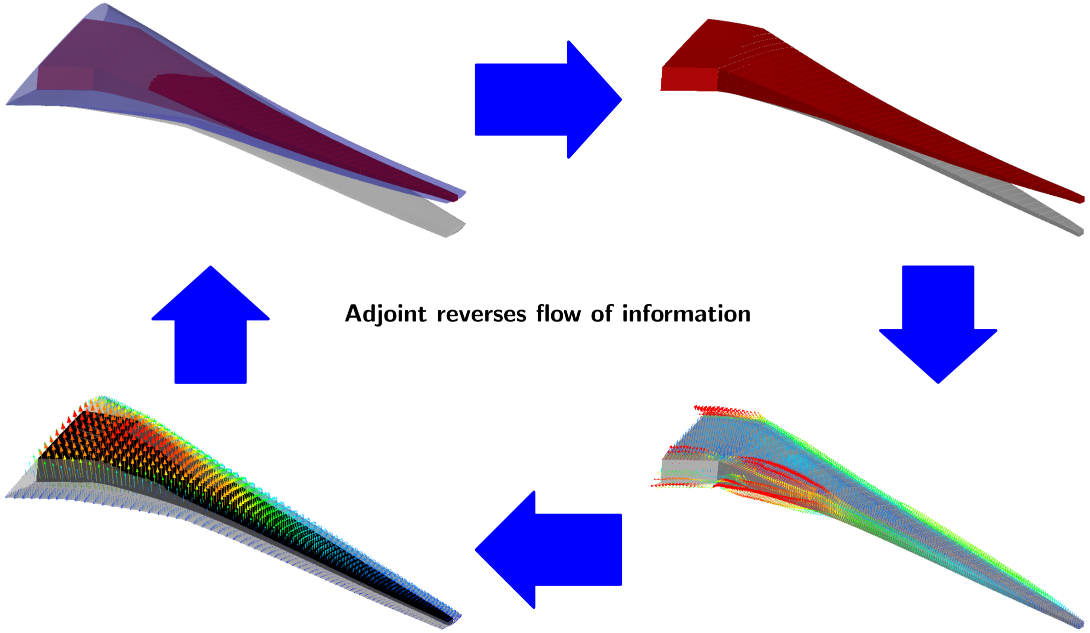
Why Adjoint-based Optimization?

- High-fidelity aeroelastic models
  - High computational cost
  - $O(10^2\text{-}10^4)$ design variables
- Gradient-free methods:
  - Global optimum
  - Scale poorly with # of design variables
- Gradient-based methods:
  - Local optimum
  - Better scaling w.r.t. # of design variables
  - Gradient calculation:
    - Finite-difference or tangent method:
      - $O(n)$ w.r.t. # of design variables
      - Essentially independent of # of functions of interest
    - Adjoint method:
      - **Essentially independent of # of design variables**
      - $O(n)$ w.r.t. # of functions of interest



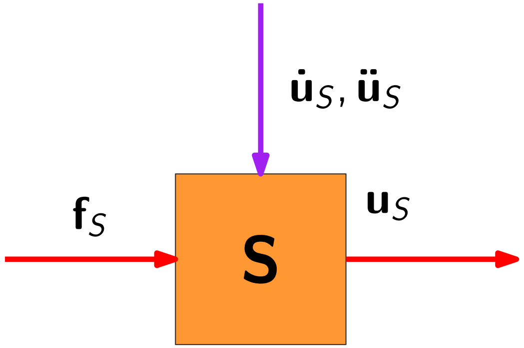
Discrete Adjoint Method Summary

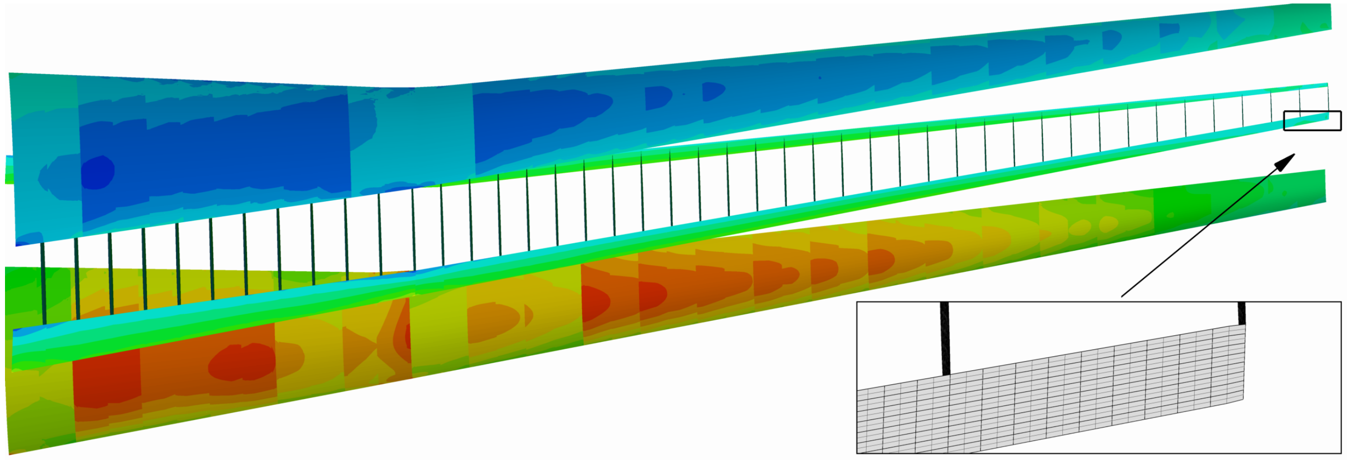
1. Solve the discretized governing equations and evaluate the function
2. Solve the adjoint equations
   - Linear system
   - Reverses the propagation of information
3. Assemble the gradient from the adjoint solution



TACS

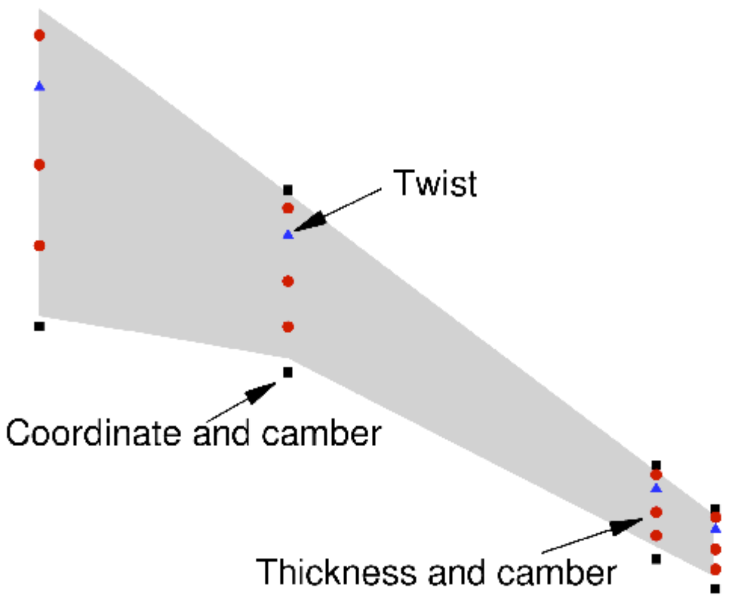
- Structural FEM from Graeme Kennedy’s SMDO Lab at Georgia Tech
- Open-source code
- Elements for geometrically linear/nonlinear analysis
- Flexible multibody dynamics
- Hand-coded discrete adjoint

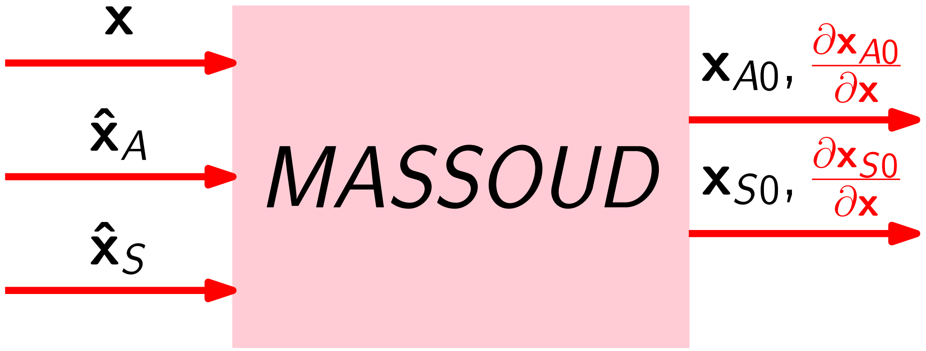


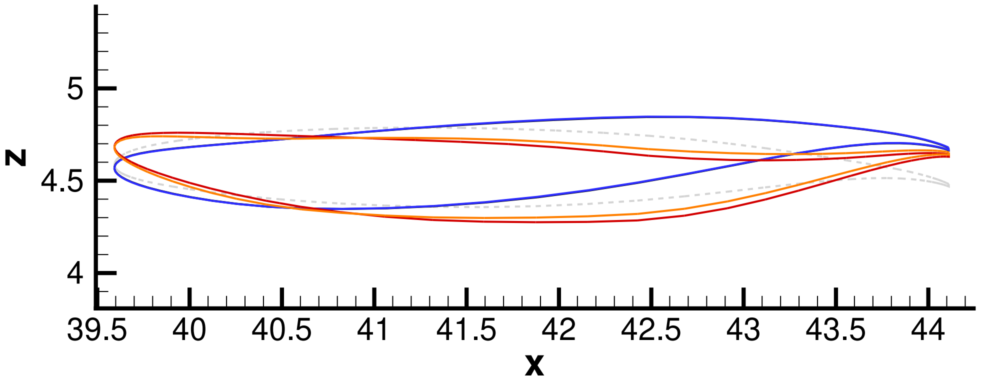


MASSOUD

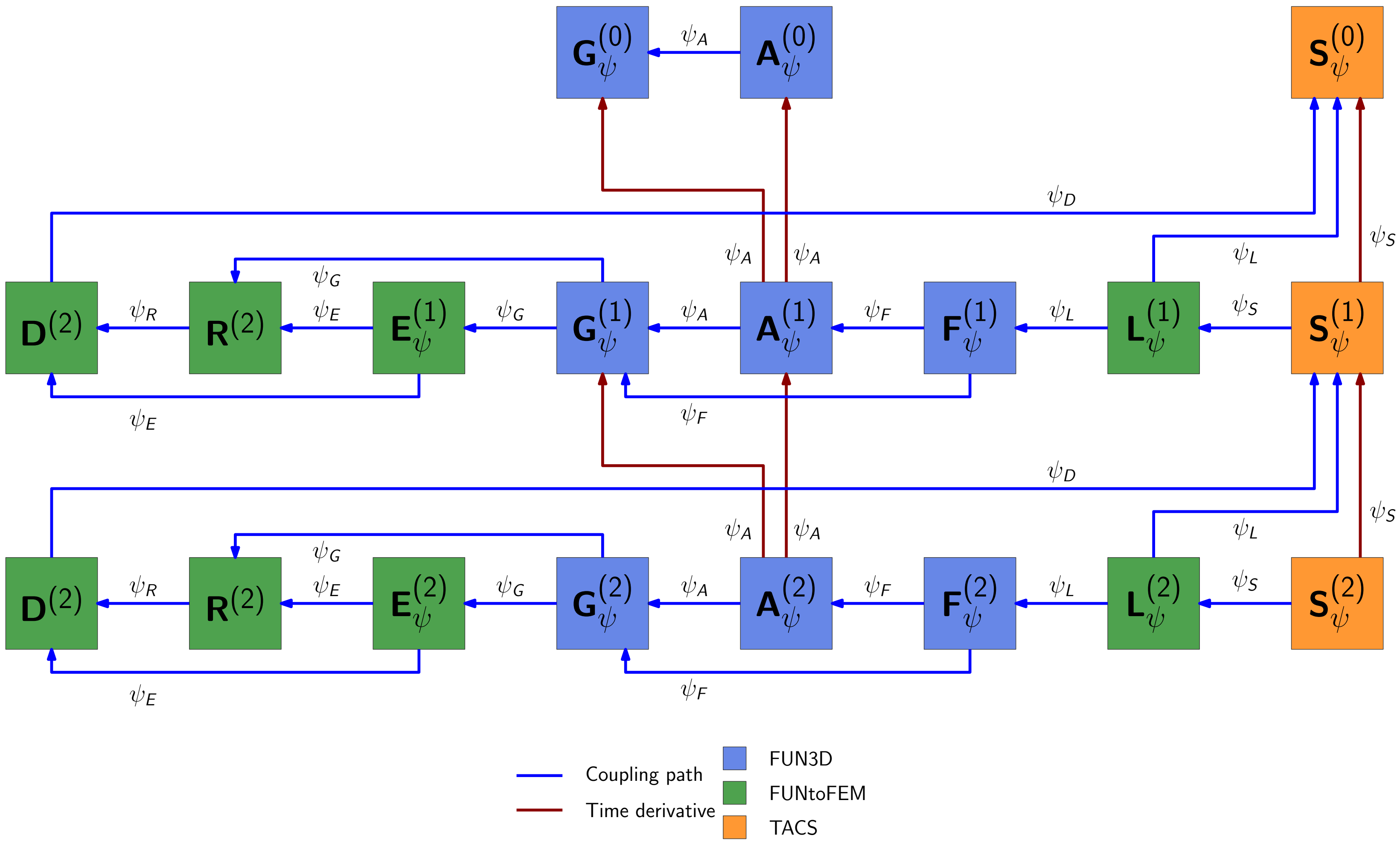
- NASA Langley shape parameterization tool
  - Based on free-form deformation (soft object animation)
  - Variables are based on standard wing design concepts
    - Camber, thickness, twist, planform coordinates
  - Parameterize aerodynamic surface and structural meshes
  - Given shape design variables:
    - Returns updated aerodynamic surface and structure meshes
    - Returns design velocities (coordinate sensitivities)





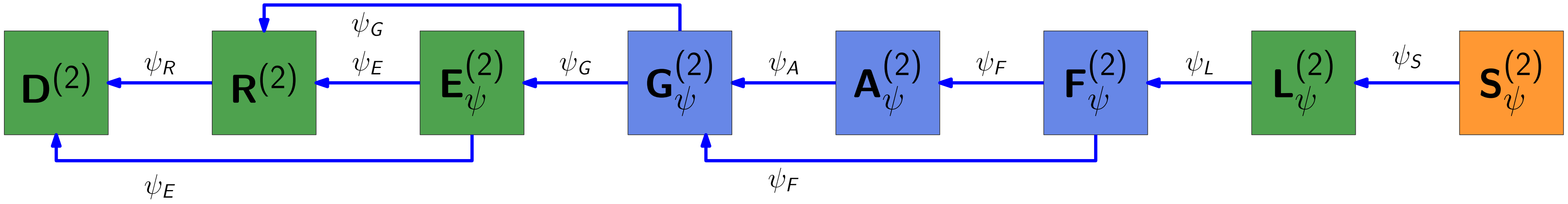


Time-accurate Coupling - Adjoint



Time-accurate Coupling - Sensitivities

$\mathbf{R}_{\psi}^{(2)}$'s contribution to gradient is $\left(\psi_R^{(2)}\right)^T \left[\frac{\partial \mathbf{R}^{(2)}}{\partial \mathbf{x}}\right]$

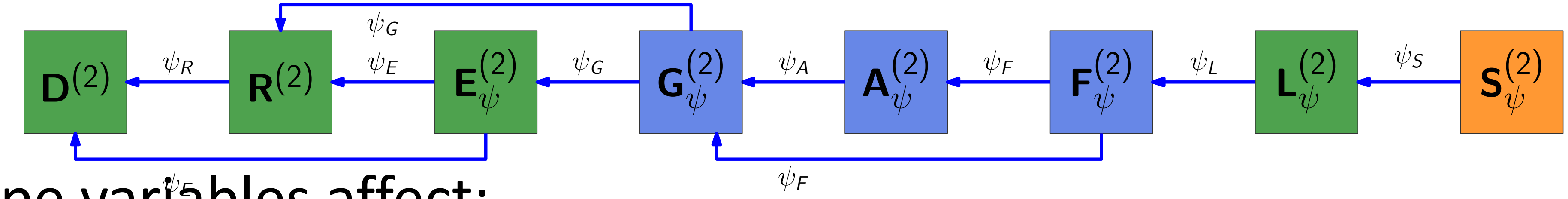


- Derivative w.r.t. aerodynamic design variable:

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \sum_{k=1}^{N} \left[\left(\psi_A^{(k)}\right)^T \frac{\partial A^{(k)}}{\partial \mathbf{x}} \right] + \left(\psi_A^{(0)}\right)^T \frac{\partial A^{(0)}}{\partial \mathbf{x}}$$

- Accumulate during reverse time marching

Shape Sensitivities



- Shape variables affect:
  - Aerodynamic surface coordinates, $\mathbf{x}_{A,0}$
  - Structural coordinates, $\mathbf{x}_{S,0}$
- Combine with design velocities using the chain rule:

$$
\begin{aligned}
\frac{df}{d\mathbf{x}} = &\left\{ \frac{\partial f}{\partial \mathbf{x}_{A,0}} + \sum_{k=1}^{N} \left[\left(\psi_D^{(k)}\right)^T \frac{\partial \mathbf{D}^{(k)}}{\partial \mathbf{x}_{A,0}} + \left(\psi_R^{(k)}\right)^T \frac{\partial \mathbf{R}^{(k)}}{\partial \mathbf{x}_{A,0}} + \left(\psi_E^{(k)}\right)^T \frac{\partial \mathbf{E}^{(k)}}{\partial \mathbf{x}_{A,0}} \right.\right. \\
&\left.\left. + \left(\psi_L^{(k)}\right)^T \frac{\partial \mathbf{L}^{(k)}}{\partial \mathbf{x}_{A,0}} + \left(\psi_G^{(k)}\right)^T \frac{\partial \mathbf{G}^{(k)}}{\partial \mathbf{x}_{A,0}} \right] + \left(\psi_G^{(0)}\right)^T \frac{\partial \mathbf{G}^{(0)}}{\partial \mathbf{x}_{A,0}} \right\} \frac{\partial \mathbf{x}_{A,0}}{\partial \mathbf{x}} \\
&+ \left\{ \frac{\partial f}{\partial \mathbf{x}_{S,0}} + \sum_{k=1}^{N} \left[\left(\psi_D^{(k)}\right)^T \frac{\partial \mathbf{D}^{(k)}}{\partial \mathbf{x}_{S,0}} + \left(\psi_L^{(k)}\right)^T \frac{\partial \mathbf{L}^{(k)}}{\partial \mathbf{x}_{S,0}} + \left(\psi_S^{(k)}\right)^T \frac{\partial \mathbf{S}^{(k)}}{\partial \mathbf{x}_{S,0}} \right] + \left(\psi_S^{(0)}\right)^T \frac{\partial \mathbf{S}^{(0)}}{\partial \mathbf{x}_{S,0}} \right\} \frac{\partial \mathbf{x}_{S,0}}{\partial \mathbf{x}}
\end{aligned}
$$

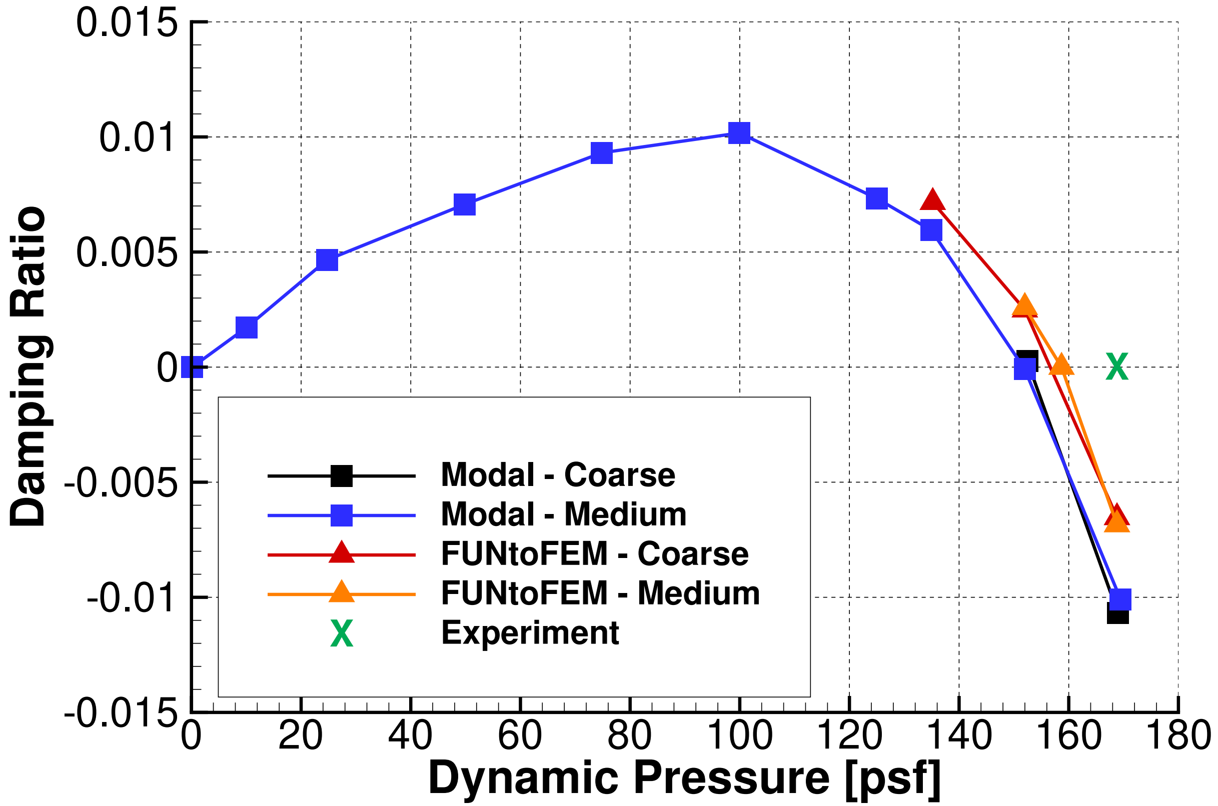
Types of Flutter Constraints

Flutter Clearance

- Constraint:
  - At q_f: $\zeta_{\min} \geq \zeta_0$
- Similar to flutter flight testing

Flutter Identification

- Additional design variable: $q_{flutter}$
- Constraints:
  - At $q_{flutter}$: $\zeta_{\min} = 0$
  - $q_{flutter} > 1.15\, q_{flight}$
- Gets the exact flutter conditions



◆ Need damping and its sensitivity regardless of constraint type

Matrix Pencil Method

- Prony series with noise term:

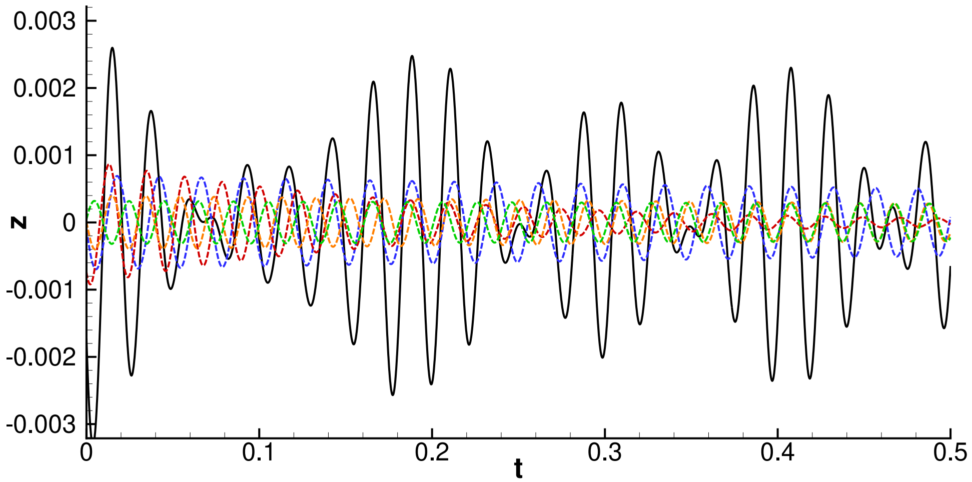
$$z_n = \sum_{k=1}^{M} c_k e^{s_k n} + w(n)$$

- Complex exponent:

$$s_k = (\alpha_k + i\omega_k)\Delta t$$

- Damping ratio:

$$\zeta_k = -\alpha_k / \omega_k$$

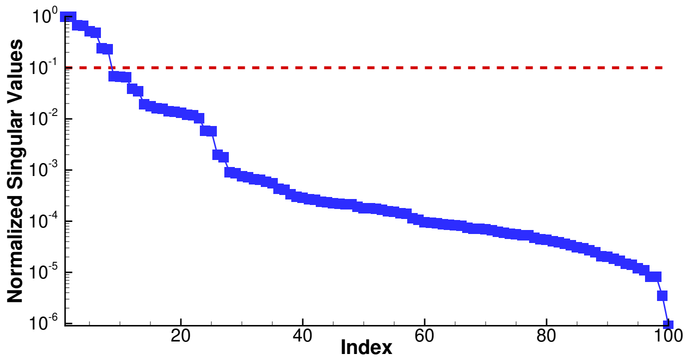


Solution Process:

1. Form the Hankel Matrix

$$\mathbf{Y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_L & y_{L+1} \\ y_2 & y_3 & \cdots & y_{L+1} & y_{L+2} \\ \vdots & \vdots & \ddots & \vdots & \\ y_{N-L} & y_{N-L+1} & \cdots & y_{N-1} & y_N \end{bmatrix}$$

2. Singular Value Decomposition (SVD) of the Hankel Matrix
   - Filter by singular values



   - Form reduced system from remaining singular vectors

3. Eigenvalues of reduced system are s_k

Constraint Aggregation

- Flutter identification constraint based on minimum damping:

$$\zeta_{min} = 0 \qquad \Rightarrow \qquad \alpha_{max} = 0$$

- Kreisselmeier-Steinhauser (KS) function
  - Differentiable approximation of the maximum exponential coefficient

$$\alpha_{max} \approx m + \frac{\ln\left[\sum_{k=1}^{M} e^{\rho(\alpha_k - m)}\right]}{\rho}$$

  - Single adjoint solution for any number of modes and frequencies

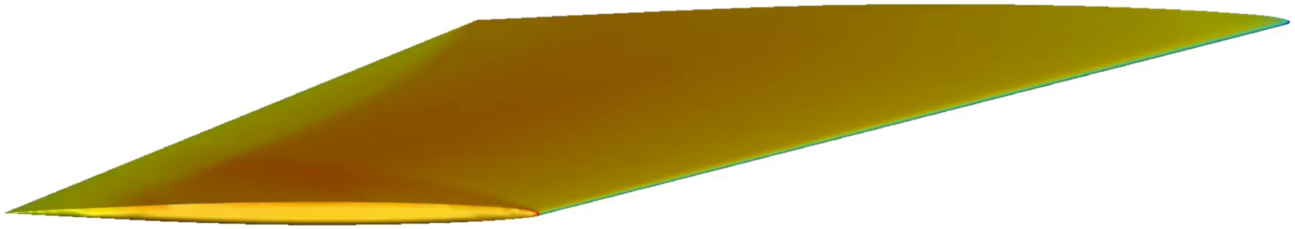
AGARD 445.6 Flutter Identification

Computational Model:

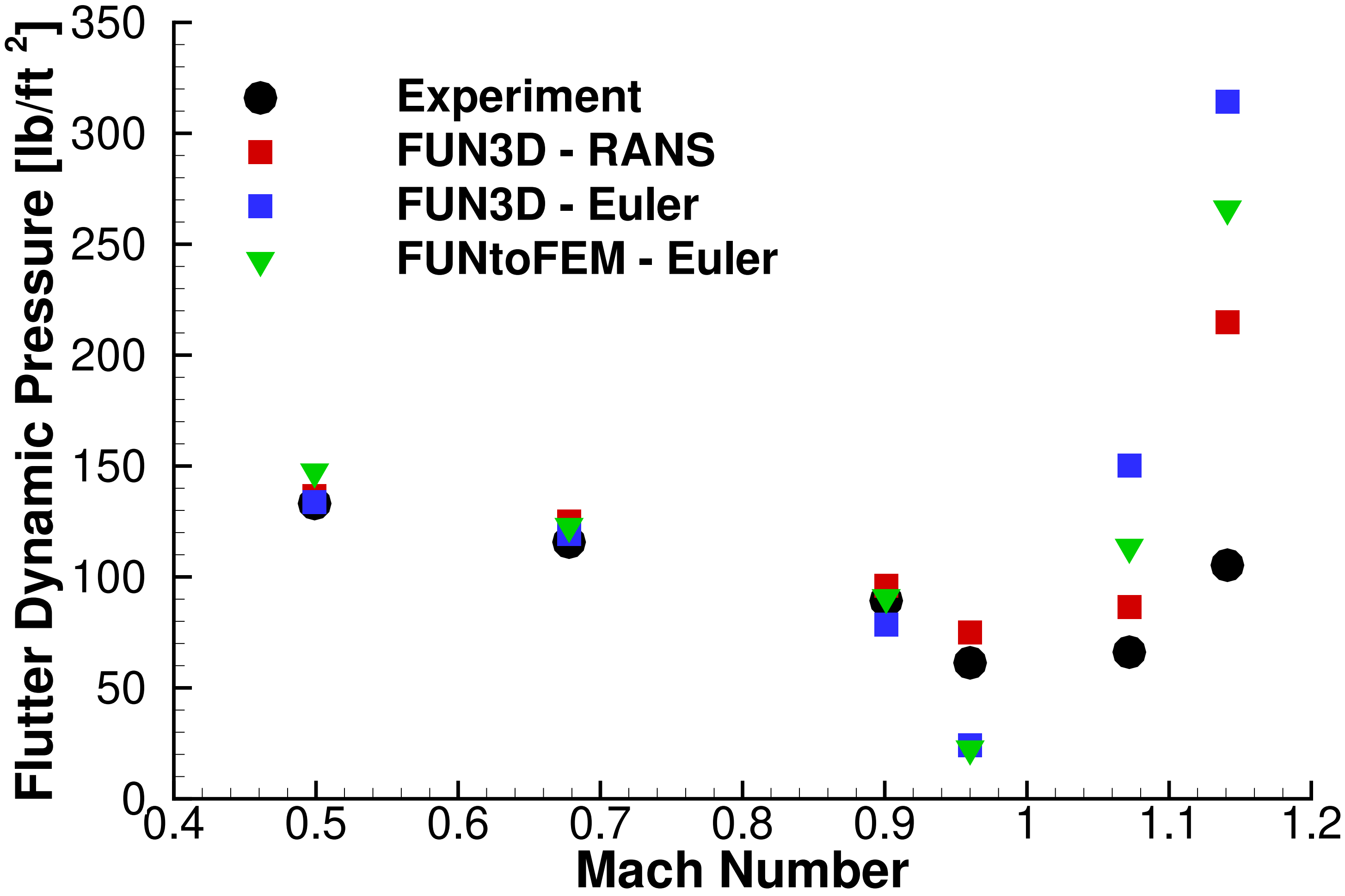
- M_∞ range: 0.499 to 1.141
- 6,000 time steps
- Structural modes excited at 125 steps
- Matrix pencil method over last 3,000 steps
- FUN3D
  - Euler, compressible simulation
  - BDF2opt integration
  - 446,584 nodes
- Modal structure (Python)
  - First 4 modes
  - BDF2 time integration

Design Problem:

- Design variables:
  Dynamic pressure, q
- Objective:
  Minimize q
- Constraints:
  $\zeta_{KS} = 0$

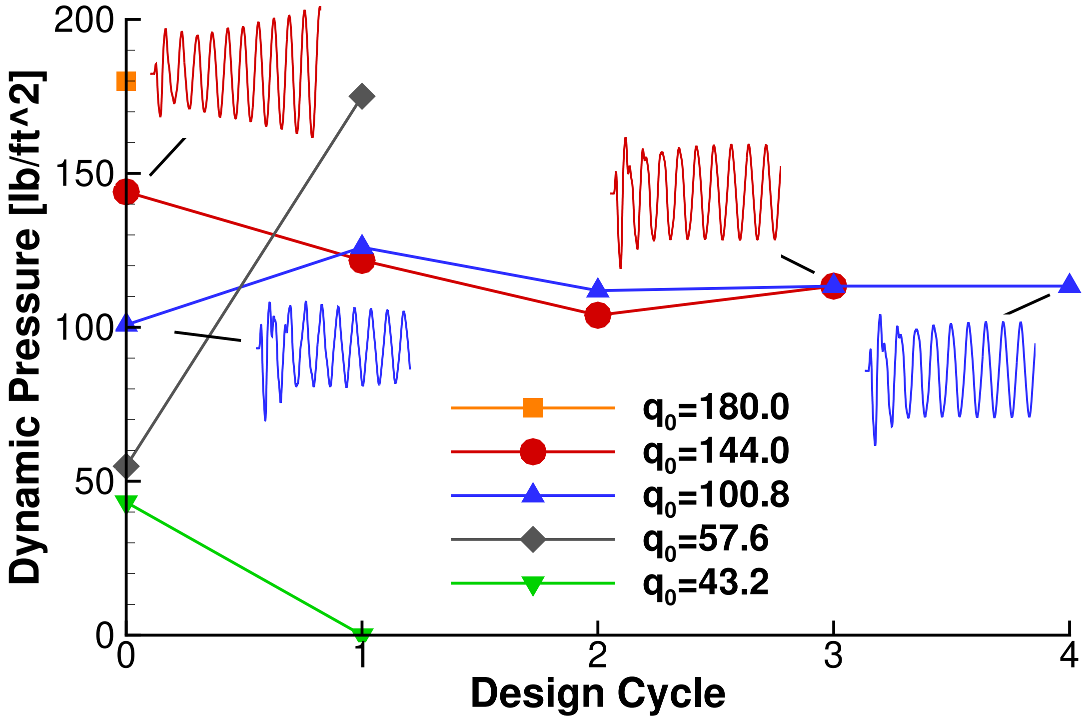


AGARD 445.6 Flutter Identification



- Converged to within 0.1 psf in 4-7 design cycles
- Coarse mesh and time step still captures the important trends

Sensitivity to initial guess for dynamic pressure:



- In a real optimization, you have the additional constraint $q_{flutter} > 1.15\ q_{flight}$
- Frequency domain methods may be more practical
  - Take advantage of periodic nature
  - Eliminate need for damping system identification

Gust Constraints - Modeling in FUN3D

Field Velocity Method:

- Gust created by modifying the grid velocity:

$$\dot{x}\hat{i}+\dot{y}\hat{j}+\dot{z}\hat{k}=(\dot{x}_0-u_g)\,\hat{i}+(\dot{y}_0-v_g)\,\hat{j}+(\dot{z}_0-w_g)\,\hat{k}$$

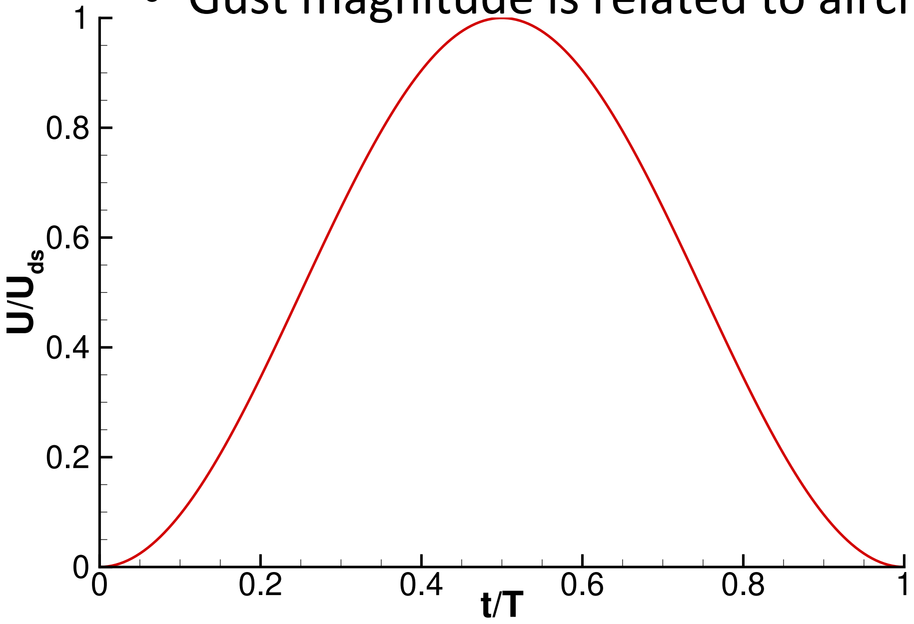
  - Only changes velocities, not the displacements
  - Not applied to surface nodes
- Gusts propagate with the freestream velocity
- Gust profile created from superposition of sines, cosines, 1-cosines, and gaussian pulses
  - options to set amplitude, frequency, start time, etc.
  - Profile is uniform in planes normal to direction of propagation

Sensitivities verification with gust model:

	Structural Thickness	Angle of Attack	Shape Variable
Lift Adjoint	-2.6070143486**5**e-05	0.00456874796182	-0.00071191638**0269**
Lift Complex	-2.6070143486**6**e-05	0.00456874796182	-0.00071191638**0507**

Gust Constraints – FAA Regulation

- FAA Advisory Circular for gust load certification:
  - Continuous gust - long duration turbulence encounters
  - Discrete gusts – single extreme turbulence event
    - 1 - cosine profile
    - Gust lengths from H=30 ft to 350 ft
    - Gust magnitude is related to aircraft weight and altitude, $U_{ref} \approx 20\text{-}56\ ft/s$



Discrete gust profile

$$U_{ds} = U_{ref} F_g \left(\frac{H}{350} \right)^{1/6}$$

Gust Demonstration – NACA 64A010

Computational Model:

- Assumed Conditions:
  - M_∞=0.8, AOA = 0°
  - h = 30,000 ft
- Gust model:
  - 1000 time steps
    - 50 steps to establish the flow field
    - ~5 gust periods
  - F_g = 0.95
  - H = 30 ft
- FUN3D:
  - Euler, compressible simulation
  - BDF2opt integration
  - 35,109 nodes
- Structure (Python):
  - Mass + vertical displacement spring
  - BDF2 integrator

Design Problem:

- Design variables:

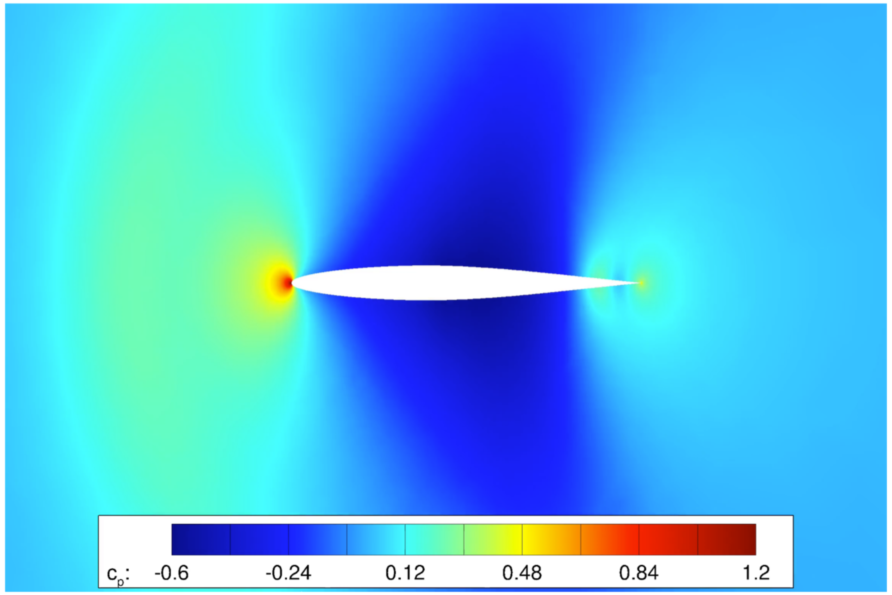
  k

- Objective:

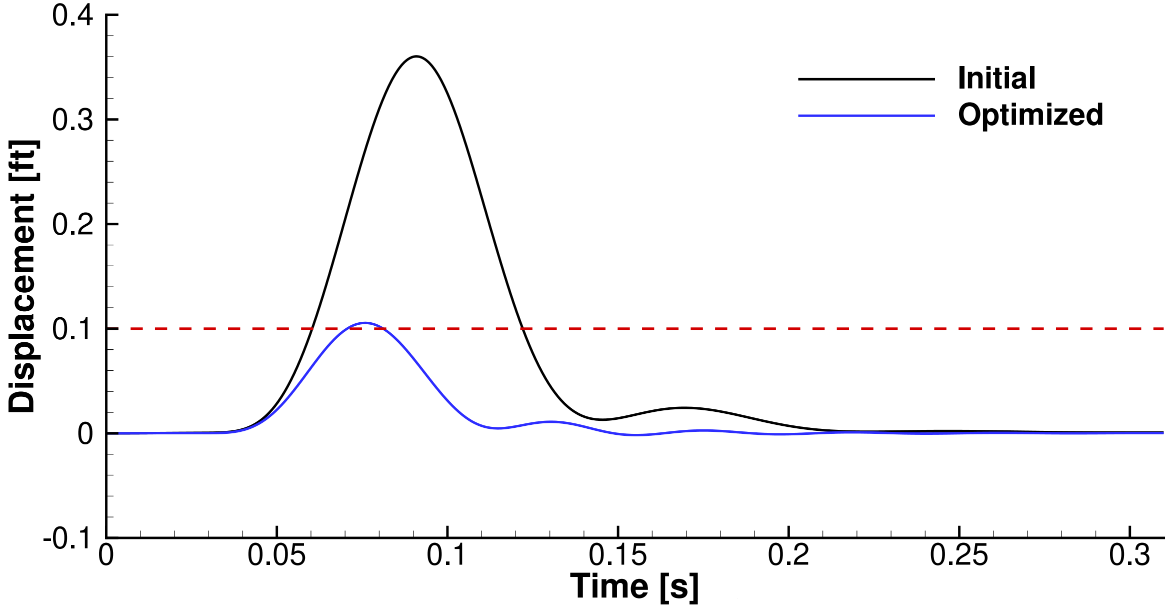
  Min k

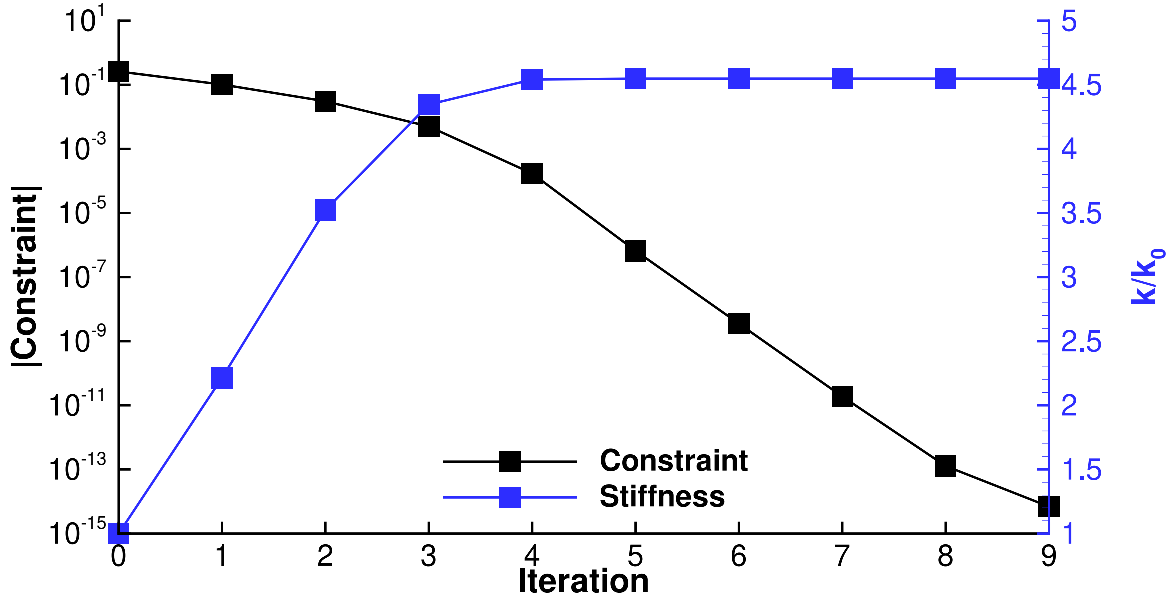
- Constraints:

  $KS(z) – 0.1 \leq 0$

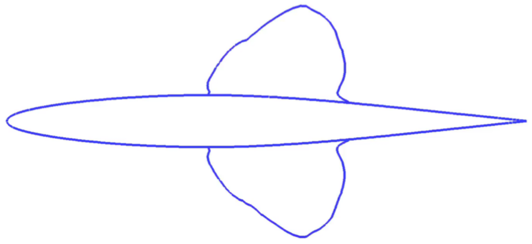


Gust Demonstration – NACA 64A010





Initial
Optimal



Sonic Line

Adjoint Method

- We have the function of interest, $f(\mathbf{x},\mathbf{q}(\mathbf{x}))$, and we want the sensitivities:

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{x}} \qquad (1)$$

- The state vector, q, is governed by the residual, $R(x,q(x))=0$:

$$\frac{d\mathbf{R}}{d\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]\frac{\partial \mathbf{q}}{\partial \mathbf{x}} = 0$$

- Rearrange:

$$\frac{\partial \mathbf{q}}{\partial \mathbf{x}} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1}\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

- Plug into *(1)*:

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial f}{\partial \mathbf{q}}\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^{-1}\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

Adjoint Method

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial f}{\partial \mathbf{q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}} \right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

- Define the adjoint vector:

$$\psi^T = -\frac{\partial f}{\partial \mathbf{q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}} \right]^{-1}$$

- Substitute to get adjoint sensitivity expression:

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \psi^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}$$

- Get the adjoint vector from the adjoint equations:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}} \right]^T \psi = -\left[\frac{\partial f}{\partial \mathbf{q}} \right]^T$$